

AD-A243 637

UNLIMITED

13505005

(2)



Report No. 91026



Report No. 91026

ROYAL SIGNALS AND RADAR ESTABLISHMENT,
MALVERN

DTIC
ELECTE
DEC 20 1991
S C D

DIAGNOSTIC RHYME TEST
STATISTICAL ANALYSIS PROGRAMS

Authors: A Sim, R Bain, A J Belyavin
& R L Pratt

UNCLASSIFIED DOCUMENT A
Approved for public release;
Distribution Unlimited

91-18555



PROCUREMENT EXECUTIVE, MINISTRY OF DEFENCE
RSRE
Malvern, Worcestershire.

August 1991

UNLIMITED

0112420

CONDITIONS OF RELEASE

305835

DRIC U

COPYRIGHT (c)
1988
CONTROLLER
HMSO LONDON

DRIC Y

Reports quoted are not necessarily available to members of the public or to commercial organisations.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

REPORT No 91026

TITLE: DIAGNOSTIC RHYME TEST STATISTICAL ANALYSIS PROGRAMS

AUTHORS: A Sim*, R Bain, A J Belyavin** and R L Pratt

DATE: August 1991

ABSTRACT

This report describes the statistical techniques and associated computer programs employed to analyse data obtained from Diagnostic Rhyme Tests (DRT), and was previously published as a CC4 Divisional Memorandum. The conduct of the DRT (used for quantifying speech intelligibility) was described in RSRE Report No 87003.

* Atomic Weapons Establishment, Aldermaston

** RAF Institute of Aviation Medicine, Farnborough

Copyright
C
Controller HMSO London
1991

Approval For	
REF. ORAL	<input checked="" type="checkbox"/>
DRT Tab	<input type="checkbox"/>
Unpublished	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



CONTENTS

1 INTRODUCTION

- 1.1 The Diagnostic Rhyme Test
- 1.2 RSRE Implementation
- 1.3 Statistical Analysis
- 1.4 Summary of Output

2 OPERATING INSTRUCTIONS

3 PROGRAM SPECIFICATIONS

- 3.1 @DRT1 Program Specification
- 3.2 @DRT2 Program Specification
- 3.3 @DRT3 program Specification

ANNEX A

- 1 Program Descriptions and Listings
- 2 Program Structure Charts
- 3 Sample program outputs
- 4 Sample control file for DRT2

1 INTRODUCTION

1.1 The Diagnostic Rhyme Test

The Diagnostic Rhyme Test (DRT) is used extensively for assessing the intelligibility of military communications systems and has become an accepted NATO standard for testing Linear Predictive Coders. A discussion of the acoustic and phonetic research on which the test is based is beyond the scope of this report, consequently only a brief outline of the test is given here.

The DRT vocabulary comprises ninety-six minimally contrasting rhyming word pairs, the initial consonants of which differ only by a single acoustic feature, or attribute. There are six such attributes; Voicing, Nasality, Sustention, Sibilation, Graveness and Compactness. As an example, the attribute voicing is present when the vocal cords are excited; in the word pair "veal-feel", the consonant "v" is voiced, but the consonant "f" is unvoiced.

The DRT is implemented using the following procedure. The 192 word vocabulary is first recorded by a set number of talkers, usually 5 or 10. A total of thirty different preordained word orders are available to eliminate possible learning effects by listening subjects. The microphone used to make the recordings and the ambient acoustic environment are selected according to the proposed operational deployment of the communication system under test. The recordings are then processed by passing them through the actual communication system (or a laboratory simulation if this is not possible) and re-recorded. The processed word lists are then presented aurally to a panel of listeners via the appropriate earphone transducer which is usually contained in a helmet, headset or handset. At the same time the listeners are also presented the appropriate word pair visually on a VDU. They then select the word they thought they heard by pressing one of two buttons. The intelligibility score is expressed as a proportion of the number of words correctly identified by the listeners, with the scores adjusted for chance so that a subject guessing will score 0% rather than 50%.

To ensure adequate stability of the results, tests are conducted using not less than five talkers and eight listeners. Each of the talkers utters the 192 word vocabulary at a rate of one word every 1.33 seconds; a five talker test therefore lasts approximately twenty-five minutes. Male and female subjects are recruited from the local population and paid for their participation as listeners in the tests which are normally conducted three mornings a week. The implementation is very close to that employed by US testing agencies, thereby allowing the exchange of speech material between the US and the UK for comparative assessment.

1.2 RSRE Implementation

The DRT system developed at RSRE involves a network of British Broadcasting Corporation (BBC) microcomputers comprising 12 listener

stations, a file-server and a master station, the listeners' VDUs being installed in two high-noise chambers. The listeners indicate their choice of word by pressing either the left or right button on a box connected to their microcomputer and the responses are logged at the master station. At the end of a listening session the results for all the listening subjects are amalgamated and stored in a text file known as a "condition" file. This file contains header information describing the test environment and blocks of user responses; each being signified by a '1' for a correct response and a '0' for an incorrect response.

The results can be analysed locally on a BBC microcomputer by running the program DRTSTAT, however the full package is available on a DEC VAX hardware in N building South Site. The terminal in the Acoustic Laboratory connected to the VAX is a BBC microcomputer running terminal emulation software. This means that the results held on a BBC formatted disc can be sent down the communications link to create text files on the VAX, which then act as input to the Analysis of Variance program, DRTSTAT.

1.3 Statistical Analysis

The DRT statistics package consists of three programs namely DRTSTAT, DRTFILE and DRTEDIT, which are run by typing @DRT1, @DRT2 and @DRT3 respectively at the VAX terminal.

@DRT1 takes a previously stored condition file and produces statistical analysis of the condition. It also stores the basic statistical (listener and talker identification, overall mark, attribute and individual listener scores) from every DRT experiment and stores the results in the file CONDMEANS.DAT.

@DRT2 reads CONDMEANS.DAT and takes user selected conditions and results from the analysis carried out by @DRT1 and reformats them for subsequent analysis by the program PANOVUN. This Analysis of Variance computer program is part of the suite of statistical analysis software supplied by the Institute of Aviation Medicine at Farnborough and is available on the VAX.

@DRT3 provides a facility to list and edit the running file of basic statistics updated by @DRT2.

1.4 Summary of Output from DRTSTAT

The statistical information output from the program was tailored to suit the particular requirements of the DRT experiments. This section describes the rationale behind the selection of the output. See Annex A for a sample printout.

1.4.1 Basic Statistics

The final result to be output is the overall mean score for the particular condition being tested along with the standard error of listeners scores when meaned over talkers. However it is also useful to be able to see, in tabular form, the performance of each individual listener/talker combination. This is output under the heading Individual Scores.

As has already been mentioned, the DRT tests the condition for six major attributes of speech. An output is therefore required which will show condition performance when each attribute was present and absent in the utterance. This is output under the heading Attribute Scores.

1.4.2 Analysis of Variance

Two way Analysis of Variance is designed to show the main and interactive effects of two factors, each factor having more than one level. In the case of DRT it was decided that the two main factors were Listeners and Talkers, with the list parts one and two being the replications. If the talker/listener interaction is shown not to be a significant effect then the Newman-Keuls test can be applied to listeners and talkers separately.

1.4.3 Tukey Test for Nonadditivity

If the experiments were to be carried out on a single list (i.e. without replication), then the Tukey test for nonadditivity would be used to test the presence of a Talker X Listener interaction. If the test does not indicate the presence of an interaction, then the analysis of the main effects is carried out as described in the preceding section.

2

Note:

@DRT2 will display messages as the required files are generated and allow immediate running of PANCVUN if desired.

```
@DRT1  Enter Condition Filename  Enter the name of the
                                     condition for analysis in
                                     the form CND9999
```

@DRT2	Enter Control Filename	Enter the name of the file containing the control information. (See Annex A.4 for example)
-------	------------------------	--

```
@DRT3      1  Display Index      Select option required
           2  Delete an Entry
           3  End
```


3 PROGRAM SPECIFICATIONS

3.1 @DRT1 Program Specification

It was decided that a total of five sets of analysis be carried out on the results of the DRT tests and their implementation along with a basic description is described below.

3.1.1 Tukey Test for Nonadditivity

If there is only one observation in each cell of a $p \times q$ factorial experiment there can be no within-cell variation and hence no direct estimate of experimental error. The following two models may be postulated to underlie the observed data:

$$(1) \quad X_{ij} = \mu + \alpha_i + \beta_j + \epsilon_{ij}$$

$$(2) \quad X_{ij} = \mu + \alpha_i + \beta_j + \alpha\beta_{ij} + \epsilon_{ij}$$

In (1) no interaction effect is postulated; hence all sources of variation other than main effects are considered to be part of the experimental error.

In (2) an interaction term is postulated. This interaction term may be considered as a measure of nonadditivity of the main effects.

Tukey developed a test applicable to the case in which there is a single observation per cell. The test is called a test for 'nonadditivity' and its purpose is to help in the decision between models (1) and (2). In his approach to the problem, Tukey starts with the model:

$$(3) \quad X_{ij} = \mu + \alpha_i + \beta_j + \lambda \alpha_i \beta_j + \epsilon_{ij}$$

where $\alpha_i \beta_j$ is the product of the main effects and λ is a regression coefficient. Here the product term is that part of the interaction which can be expressed as the product of main effects. A test in the hypothesis $\lambda = 0$ is equivalent to a test on the hypothesis that the product terms of this form do not contribute to the prediction of the X_{ij} .

In applying Tukey's test for nonadditivity to a $p \times q$ factorial experiment with one observation per cell, the analysis of variance takes the form:

Source of Variation	SS	d.f.	MS
A	SSa	p-1	MSa
B	SSb	q-1	MSb
Residual	SSres	(p-1)(q-1)	
Nonadditivity	SSnonadd	1	MSnonadd
Balance	SSbal	(p-1)(q-1)-1	MSbal

where $SS_{bal} = SS_{res} - SS_{nonadd}$
and $SS_{rea} = SS_{total} - SSa - SSb$

The test for nonadditivity is given by:

$$F = MS_{nonadd} / MS_{bal}$$

When this F ratio exceeds the corresponding critical value, the hypothesis that model (1) (ie $\lambda = 0$) is appropriate is rejected. Tukey's test for nonadditivity is sensitive only to one source of nonadditivity - that associated with a component of interaction represented by $\alpha_i \beta_j$.

For full description of test see Winer - 'Statistical Principles In Experimental Design' pp 394-397.

Assuming a matrix of n listeners (i) by m talkers (j) and results X_{ij} , the equations are as follows:

$$SSa = (1/m) \cdot \sum_{i=1}^n (\sum_j X_{ij})^2 - (1/(n \cdot m)) \cdot \left(\sum_i \sum_j X_{ij} \right)^2$$

$$SSb = (1/n) \cdot \sum_{j=1}^m (\sum_i X_{ij})^2 - (1/(n \cdot m)) \cdot \left(\sum_i \sum_j X_{ij} \right)^2$$

$$SS_{total} = \sum_i \sum_j X_{ij}^2 - (1/(n \cdot m)) \cdot \left(\sum_i \sum_j X_{ij} \right)^2$$

$$SS_{res} = SS_{ab} = SS_{total} - SSa - SSb$$

$$SS_{nonadd} = \left(\left(\sum_i c_i d_i \right)^2 \right) / \left(\left(\sum_i c_i^2 \right) \cdot \left(\sum_j d_j^2 \right) \right)$$

$$SS_{bal} = SS_{res} - SS_{nonadd}$$

$$MSx = (ssx) / (d.f.x)$$

Computational symbols

$$(6) = 1/c \sum_{j=1}^r \left(\sum_{i=1}^c x_{ij} \right)^2 \quad (7) = 1/r \sum_{i=1}^c \left(\sum_{j=1}^r x_{ij} \right)^2$$

$$(8) = 1/(r \cdot c) \left(\sum_i \sum_j x_{ij} \right)^2 \quad (9) = \sum_i \sum_j x_{ij}^2$$

$$(10) = 1/c \sum_{j=1}^r x_{ij} \quad (10) = 1/(r \cdot c) \sum_i \sum_j x_{ij}$$

$$(12) = 1/r \sum_{i=1}^r x_{ij} \quad (13) = \sum_j (7) \cdot (6)$$

Then:

$$SS_{listeners} = (6) - (8)$$

$$SS_{stalkers} = (7) - (8)$$

$$SS_{total} = (9) - (8)$$

$$SS_{res} = SS_{total} - SS_{listeners} - SS_{stalkers}$$

3.1.2 Two Way Analysis of Variance

This analysis allows us to examine the combined effect of the two factors 'talkers' and 'listeners' ie the interactive effect. Replications in this case refer to parts 1 and 2 of the talker list. The form of the Analysis of variance is as follows:

Source	SS	d.f.	MS	F
Rows	SSr	(r-1)	SSr/(r-1)	MSr/MSerror
Columns	SSc	(c-1)	SSc/(c-1)	MSc/MSerror
RxC	SSrc	(r-1)(c-1)	SSrc/(r-1)(c-1)	
MSrc/MSerror				
Error	SSerr	rc(n-1)	SSerr/rc(n-1)	
Total	SStotal	rcn-1		

Computation of the various values is as follows:

	T1 Tc	$R_i = \sum_j \sum_k X_{ijk}$
L1		•
•		•
•		•
•	X_{ijk}	•
•		•
•		•
Ln		•

$$C_j = \sum_i \sum_k \sum_l \dots \dots \dots G = \sum_i \sum_j \sum_k$$

Computational symbols

- (1) = G^2/rcn (2) = $\sum_i \sum_j \sum_k X_{ijk}^2$
(3) = $(\sum_i R_i^2) / cn$ (4) = $(\sum_j C_j^2) / rn$
(5) = $\left[\sum_i \sum_j \left(\sum_k X_{ijk} \right)^2 \right] / n$ ie square each cells total and sum these:

Then:

$$\begin{aligned} SSr &= (3) - (1) \\ SSc &= (4) - (1) \\ SSrc &= (5) - (3) - (4) + (1) \\ SSerror &= (2) - (5) \\ SStotal &= (2) - (1) \end{aligned}$$

3.1.3 Newman-Keuls test

The Newman-Keuls test for the comparison of treatment means uses the 'Studentised Range Statistic', Qr. where r is the number of steps separating two means on an ordered scale.

Further explanation of the test will be given in terms of the DRT experiment involving six talkers and weight listeners. We shall for the moment consider the talkers, and investigate the difference between their means.

	T1	T2	T3	T4	T5	T6
L1						
L2						
.						
.						
.						
L8						
	$\bar{T}1$	$\bar{T}2$	$\bar{T}3$	$\bar{T}4$	$\bar{T}5$	$\bar{T}6$
posn	a	b	c	d	e	f

Fig 3.1

The Talker means should first be ranked $\bar{T}(1) \rightarrow \bar{T}(6)$ where $\bar{T}(1)$ is the smallest and $\bar{T}(6)$ the largest. $\bar{T}(6)$ is defined as being six steps from $\bar{T}(1)$. In general the number of steps between the ordered means is $j-i+1$.

Thus, $Q6 = \{ \bar{T}(6) - \bar{T}(1) \} / \sqrt{MS_{error}/n}$

$n = 8$ here (16 if 2 reps)

There is only one Q6 statistic. There are however two Q5 statistics, namely:

$$Q5 = \{ \bar{T}(6) - \bar{T}(2) \} / \sqrt{MS_{error}/n}$$

and

$$Q5 = \{ \bar{T}(5) - \bar{T}(1) \} / \sqrt{MS_{error}/n}$$

Critical values for Qr are obtained from tables of the studentised range statistic by setting r equal to the range. For example, the critical value for Q6 in a 0.01-level test is Q.99 (6,f) where f is the degrees of freedom for MS_{error}.

In making a large number of test it is more convenient to obtain a critical value for the difference between two means:

$$\bar{T}(6) - \bar{T}(1) = QR \sqrt{MS_{error}/n}$$

The six talker means are designated by the symbols 'a' through to 'f',

and a new table is constructed with the talker means in ranked order across the top.

\bar{T}_j	\bar{T}_3	\bar{T}_3	\bar{T}_3	\bar{T}_3	\bar{T}_3	\bar{T}_3	'say'	
	c	a	d	b	f	e	r	val
c	-	$\bar{T}_1 - \bar{T}_3$	$\bar{T}_4 - \bar{T}_3$	$\bar{T}_2 - \bar{T}_3$	$\bar{T}_6 - \bar{T}_3$	$\bar{T}_5 - \bar{T}_3$	6	x1 , say
a		-	$\bar{T}_4 - \bar{T}_1$	$\bar{T}_2 - \bar{T}_1$	$\bar{T}_6 - \bar{T}_1$	$\bar{T}_5 - \bar{T}_1$	5	x2
d			-	$\bar{T}_2 - \bar{T}_4$	$\bar{T}_6 - \bar{T}_4$	$\bar{T}_5 - \bar{T}_4$	4	
b				-	$\bar{T}_6 - \bar{T}_2$	$\bar{T}_5 - \bar{T}_2$	3	
f					-	$\bar{T}_5 - \bar{T}_6$	2	

$$\text{where val} = Q.99 (r, f) \sqrt{MS_{\text{error}}/n}$$

Fig 3.2

In general the entry in column j , row i is $\bar{T}_j - \bar{T}_i$. Only the entries shown in this table need be computed. It should be noted that entries on the same diagonal on the table have the same number of steps between each pair of means being compared.

There is a prescribed sequence in which tests on differences between ordered means should be made:

- 1 The first test is made on the difference in the upper right-hand corner. This is the difference for which r has its maximum value, which is in the case $r = 6$. Let the critical value for a 0.01-level test be x_1 . If $\bar{T}_5 - \bar{T}_1$, ie $\bar{T}(6) - \bar{T}(1)$ does not exceed this value one need not carry out any further tests since, if there is no significant difference between the means furthest apart from each other, there will clearly be no significant difference between any pair of means closer together.
- 2 Tests are now made on all differences for which $r = 5$. If any of the difference values on the diagonal for which $r = 5$ do not exceed the critical value, say x_2 , no further tests are made on the triangular region defined by the rows and columns of which this particular difference forms the upper right-hand corner. For example, if $\bar{T}_5 - \bar{T}_1 < x_2$ there would be no need to test the following:

$\bar{T}_4 - \bar{T}_1$	$\bar{T}_2 - \bar{T}_1$	$\bar{T}_6 - \bar{T}_1$	
	$\bar{T}_2 - \bar{T}_4$	$\bar{T}_6 - \bar{T}_4$	$\bar{T}_5 - \bar{T}_4$
		$\bar{T}_6 - \bar{T}_2$	$\bar{T}_5 - \bar{T}_2$
			$\bar{T}_5 - \bar{T}_6$

Other entries on the diagonal, though, should be tested, eg $\bar{T}_6 - \bar{T}_3$, and the same eliminating procedure carried out where necessary. Thus we work through on the entries on the table given above.

The schematic representation of the significant differences between means of talkers take the form.

c a d b f e

Fig 3.3

The letters underlined by a common line are not significantly different to each other, ie there is no significant difference between \bar{T}_5 , \bar{T}_6 , \bar{T}_2 and \bar{T}_4 , nor between \bar{T}_3 , \bar{T}_1 , \bar{T}_4 .

3.1.4 Attribute Scores

The DRT experiments are designed to examine the six major attributes which apply to consonant sounds of speech. These are:

Voicing
 Nasality
 Sustention
 Sibilation
 Graveness
 Compactness

A table of mean and standard error of listeners scores is therefore required in the case where each of the six attributes were separately present and absent. Also included in the table is the mean and standard error of the combined present and absent scores.

3.1.5 Individual Scores and Overall DRT Score

The basic statistics of individual listeners scores are required for detailed examination of listeners performance and to allow

interpretation of, eg the Newman-Keuls test. The results are looked at from two 'directions', ie listener's score over talkers, and talker score over listeners.

The final statistic required is the overall DRT score and standard error.

To allow for archiving of results for further analysis certain values are stored in an indexed file, the key field being the condition number. The values stores are as follows:

- Condition name
- Listener's ID
- Listener's mean scores
- Listener's attribute scores
- Talker results
- Talker's ID
- Number of talkers

3.2 @DRT2 Program Specification

In order to utilise the results saved by @DRT1, a program is required which will manipulate selected results into a form which can be processed by the statistics program PANOVUN. This format, known as EDA, is an Institute of Aviation Medicine standard.

Inputs:

- 1 Name of EDA file to be created.
- 2 Attributes required from conditions.
- 3 Conditions to be considered.
- 4 Particular listeners of interest.
- 5 Format of EDA file.
- 6 Design statements required for PANOVUN
- 7 List of all possible listener idents.

Outputs:

- 1 Data output files.
- 2 Messages to screen.

3.3 @DRT3 Program Specification

As the basic statistics from @DRT3 are automatically stores in a running file, it is necessary to be able to examine that file to see what Condition records are held. It is also necessary to be able to delete records in order to stop the file from becoming too large and overloaded with outdated information.

A program is required which will satisfy these requirements and also be written in such a way that will allow further editing facilities to be added in the light of changes in policy.

Input:

Either select Index of Conditions or Condition for deletion.

Output:

Either list of Condition numbers or notification of Condition deleted.

A1 PROGRAM DESCRIPTION

A1.1 @DRT1 Program Description

The command file prompts for the Condition file, copies it to DRTRES and runs the program DRTSTAT. As can be seen by reference to the structure chart, the program consists of eleven procedures which are described as below:

INITIALISE

- 1 Set 'Present or Absent' array to zero.
- 2 Set listener means array to zero.
- 3 Set listener ID array to '----' and talker ID array to '----'.

UPDATECONDMEANS

- 1 Collect condition name, listener and talker IDs, attribute scores, overall means, and talker results.
- 2 Open file.
- 3 If not stored by a previous run, store results using condition as key.
- 4 Close file.

READSEQFILE

- 1 Each DRT list has a different sequence in which the particular attribute being tested is present or absent. These sequences are read in from file into array SEQ.

READNKCONS

- 1 Read the file of constants required by the Newman-Keuls test.

READWRITE

- 1 Read the number of lines in parameter from DRTRES and output to printer.

READRESFIL

- 1 Reset DRTRES for reading
- 2 Read experiment ID
- 3 Read version number and output necessary number of lines.
- 4 Build up array of talker IDs.

5 If DYNASEQ was used then for each of the presentations in cnd file, calculate which SEQ applies.

6 Build up array of listener IDs.

7 Now that the first page of information has been read, the blocks of listener response can be read. The nesting is listeners within parts within talkers. Two results are calculated; the attribute scores and the overall scores. Attributes are only calculated if DYNASEQ is used. In which case, if a correct response is read (ie a '1') the sequence array is checked to find if the attribute was present or absent and the correct element in PORA incremented. The variable ATT cycles through the six attributes. Both the attributes and the overall results have the algorithm applied which gives a score of zero for completely random responses.

SINGLEREPS

1 Produces a matrix with the parts average for the Tukey test.

SETUNLNS

1 Build up the array ULNS which gives the number of dashes required for up to 12 positions from a left margin, ie pos 1 requires 1 dash, pos 3 requires 5. This is required when deciding how many elements in the Newman-Keuls test have to be underlined.

NKCALCS

1 Read in constants for calculating critical values.

2 Build up the array POSITIONARR.

3 If looking at talkers, CONV will contain numoflisners and vice-versa.

4 Sort the array of means (either listener or talker) into lowest to highest. Also sort the corresponding position in POSITIONARR.

5 Build up the values corresponding to Fig 3.2.

6 If no significant difference as in 3.3.1 the print out ordered means else ...

7 Carry out comparisons and calculate line lengths required (Fig 3.3).

8 Printout.

STATSONATTS

1 Calculate attribute means, sum of squares, and standard error.

2 Write results for attributes.

ANOVAR

1 Call READRESFIL; SINGLEREPS;

2 Calculate components required as in 3.1 and 3.2.

3 Write results for Tukey and ANOVA,

4 Calculate listener and talker means and call NKCALCS.

5 If DYNASEQ was used then call STATSONATTS.

6 Calculate individual means, grandmean, standard deviation, standard error for both listeners over talkers and talkers over listeners.

7 Write results for basic stats.

A1.2 @DRT2 Program Description

The command file prompts for the control file to be used and the name of the DRTFILE output file to be used as input to PANOVUN. It then runs DRTFILE followed by PANOVUN if desired. Whilst running, DRTFILE will report any errors it sees in the input data but whenever possible will allow the output file to be created. This is to allow the operator to ignore apparent errors for a particular requirement. As can be seen on the structure chart, DRTFILE consists of nine procedures and two functions as follows:

Function INTTOCHR

Takes integers in the range 0-99 and turns them into strings two characters long.

Function CHRTOINT

Takes strings of two characters and returns integers in the range 0-99.

Function TOCHARS

This function simply turns the real numbers stores on the file CONDMEANS into characters suitably formatted for the output text file.

1 If a score of 100% is encountered then write '100'el%e..

2 Check for sign to be inserted in first element of field.

3 Split the number into its character components.

4 Pack in into a string.

ENTERNAMES

1 Read the listener ids at the start of each data block and check to ensure that each 4 character name exists in the set of listener ids at the end of the control file. Build up an index of the listener id numbers.

UPDATE

Using recursion, update the parameters with the nested values for output.

OUTPTNESTING

Call UPDATE to find the latest values for the nestings and output them.

TOEDAFORM

Read the control file to find the format required for the EDA type file required by PANOVUN. After checking for errors and extracting the required information, write the header information to the output file. Cycle through the temporary file created earlier reading the listener names, extracting the required listeners results, checking for completeness, sorting and printing out the results to the output file. Finally add the required number of -1's.

DOEDAINPT

Read that part of the control file which contains the design statements for PANOVUN and write them to DRTEDA.

OUTDATA

1 Unpack string and pick our 3rd component.

2 Apply a hashing function to obtain a corresponding number for the attribute.

3 Relate the has number to the required value in PAORM and write the value after applying TOCHARS.

SELECTDATA

1 Write condition number and listener IDS.

2 Step through the string of attributes required and call OUTDATA for each until complete.

RETRIEVEREC

- 1 Locate required condition using cnd number as key.
- 2 If data located, generate attribute means and call SELECTDATA to pick out required elements.

DECODESTRGS

- 1 Find length of user input strings for conditions and attributes required.
- 2 Pick out each condition number and..
- 3 If a range has been specified, work out what each number would be in the range and call RETIEVEREC for each else..
- 4 Call RETRIEVEREC for condition number.

Main program

- 1 Open files.
- 2 Read input file and build up strings of attributes required and conditions.
- 3 Issue running message.
- 4 Call DECODESTRINGS.
- 5 If any of the conditions required do not exist then report and stop else...
- 6 Call TOEDAFORM to take DRTTMPFL1 and reformat it to DRTTMPFL2.
- 7 If any missing values exist then report.
- 8 Generate file of statements required for PANOVUN.

A1.3 @DRT3 Program Description

The command file runs the program DRTEDIT. This program only requires three procedures, ie DISPINDEX, DELENTY and DISPMENU.

- 1 DISPINDEX opens the file for sequential access, reads each entry and displays the condition number.
- 2 DELENTY opens the file for keyed access, find the required entry and deletes it from the file.
- 3 DISPMENU displays the options and the program CASE statement vectors to the required procedure.

```

DRT1.COM
$ WR:=WRITE SYSS$OUTPUT
$ WR " "
$ WR "DRT STATISTICS PROGRAM"
$ WR "-----"
$ WR " "
$ AGAIN:
$ INQUIRE FILNAM "Enter filename"
$ SET NOON
$ COPY 'FILNAM' DRTRES.DAT
$ IF .NOT. $STATUS THEN GOTO AGAIN
$ SET ON
$ PURGE DRTRES.DAT
$ RUN DRTSTAT

DRT2.COM
$ DEASSIGN/ALL
$ SET TERM/WIDTH=132
$ SET TERM/NOWRAP
$ WR:=WRITE SYSS$OUTPUT
$ WR " "
$ WR "DRT RESULTS REFORMATTER PROGRAM"
$ WR "-----"
$ WR " "
$ INQUIRE FILNAM "Enter Control Filename"
$ ASSIGN 'FILNAM' infile
$ ASSIGN DRTTMPFL2.DAT outfile
$ ASSIGN/USER_MODE 'F$LOGICAL("TT")' SYSS$INPUT
$ PURGE DRTEDA.DAT
$ PURGE DRTTMPFL2.DAT
$ PURGE DRTTMPFL1.DAT
$ RUN DRTFILE
$ INQUIRE RUNPAN "Run Panovun? (Y/N)"
$ IF RUNPAN .EQS. "N" THEN GOTO done
$ PURGE DRTRESULTS.LIS
$ ASSIGN DRTRESULTS.LIS FOR013
$ DEASSIGN outfile
$ ASSIGN 'F$LOGICAL("TT")' FOR014
$ ASSIGN DRTTMPFL2.DAT FOR015
$ ASSIGN 'F$LOGICAL("TT")' FOR016
$ ASSIGN DRTONELINE.DAT FOR017
$ ASSIGN 'F$LOGICAL("TT")' FOR018
$ ASSIGN 'F$LOGICAL("TT")' FOR019
$ RUN PANOVUN
$ TYPE DRTRESULTS
$ done:
$ DEASSIGN/ALL

DRT3.COM
$ ASSIGN/USER_MODE SYSS$COMMAND SYSS$INPUT
$ RUN DRTEDIT

```

```
PROGRAM DRTSTAT(INPUT,OUTPUT,DRTRES,DRTCONS,CONDMEANS,DYNASEQ);
```

```
(* RSRE Malvern 1985. *)
(* This program takes output from Diagnostic Rhyme Tests *)
(* produced by Multi-Listener Master and performs the *)
(* statistical tests defined in the specification. *)
(* Although designed initially to run on a DEC VAX vers4,*)
(* it has few Vax dependent features to allow easy *)
(* conversion for other PASCAL compilers. *)
```

```
CONST
```

```
    NUMOFREPS=2;
```

```
TYPE
```

```
    STATABC4=ARRAY[1..12] OF REAL;
    ARROFMEANS=ARRAY[1..12] OF REAL;
    ARROFUNLNS=ARRAY[1..12] OF INTEGER;
    STRING=PACKED ARRAY[1..7] OF CHAR;
    STROFFFOUR=PACKED ARRAY[1..4] OF CHAR;
    STROFTHREE=PACKED ARRAY[1..3] OF CHAR;
    CONDREC=RECORD
        CONDITION:[KEY(0)] PACKED ARRAY[1..7] OF CHAR;
        LNAMS:ARRAY[1..12] OF STROFFFOUR;
        LISNEROAMS:ARROFMEANS;
        ATTRIBUTES:ARRAY[1..12,1..2,1..6] OF REAL;
        TLKRESMAT:ARRAY[1..12,1..12] OF REAL;
        TNAMS:ARRAY[1..12] OF STROFTHREE;
        TLKNUM:INTEGER;
    END;
```

```
VAR
```

```
    PORA:ARRAY[1..12,1..2,1..6] OF REAL;
    NUMOFTALKERS,NUMOFLISNERS,NUMOFPRES:INTEGER;
    DRTRES,DRTCONS,DYNASEQ:TEXT;
    DONE:BOOLEAN;
    OAM,GRANDMEAN,STANDEV,STANERR,OASTANERR:REAL;
    SSROW,SSCOL,SSROWCOL,SSERR,SSTOTAL,MSERR:REAL;
    OPTION,I,J,K:INTEGER;
    RESMAT:ARRAY[1..12,1..12,1..NUMOFREPS] OF REAL;
    SINREPMAT:ARRAY[1..12,1..12] OF REAL;
    SEQ5:ARRAY[1..30,1..96,1..2] OF CHAR;
    RET:CHAR;
    UNLNS:ARROFUNLNS;
    LMEANS,TMEANS:ARROFMEANS;
    ATTMEAN:ARRAY[1..2,1..6] OF REAL;
    PAMEAN:ARRAY[1..6] OF REAL;
    NKCONS:STATABC4;
    EXPID:STRING;
    CONDMEANS:FILE OF CONDREC;
    OASTANDEV:REAL;
    LNRID:ARRAY[1..12] OF STROFFFOUR;
    LNRIDARR:ARRAY[1..4] OF CHAR;
    TLKID:ARRAY[1..12] OF STROFTHREE;
```



```

TLKIDARR:ARRAY[1..3] OF CHAR;
NVAL:INTEGER;
ATTSS:ARRAY[1..2,1..6] OF REAL;
ATTSE:ARRAY[1..3,1..6] OF REAL;
TITLSTR:PACKED ARRAY[1..66] OF CHAR;
SUBTITL:PACKED ARRAY[1..11] OF CHAR;
TITLPTR:INTEGER;
MATTSS:ARRAY[1..6] OF REAL;
SEQNUM:ARRAY[1..30] OF INTEGER;

```

```

PROCEDURE INITIALISE;
(* set arrays to 0 *)

```

```

BEGIN
  FOR I:=1 TO 12 DO
    BEGIN
      FOR J:=1 TO 2 DO
        FOR K:=1 TO 6 DO
          PORA[I,J,K]:=0;
        LMEANS[I]:=0;
        LNRID[I]:='----';
        TLKID[I]:='---'
      END;
    END;
  END;

```

```

PROCEDURE UPDATECONDMEANS;
(* add overall mean, lisners, attributes to file *)
VAR
  COND:CONDREC;
BEGIN
  WITH COND DO
    BEGIN
      CONDITION:=EXPID;
      FOR I:=1 TO 12 DO
        BEGIN
          LNAMS[I]:=LNRID[I];
          FOR J:=1 TO 2 DO
            FOR K:=1 TO 6 DO
              ATTRIBUTES[I,J,K]:=PORA[I,J,K];
            LISNEROAMS[I]:=LMEANS[I];
            TLKNUM:=NUMOFTALKERS;
            FOR J:=1 TO NUMOFTALKERS DO
              TLKRESMAT[I,J]:=SINREPMAT[I,J]
            END;
          FOR I:=1 TO NUMOFTALKERS DO TNAMS[I]:=TLKID[I];
        END;
      OPEN(CONDMEANS,
        HISTORY:=UNKNOWN,
        ORGANIZATION:=INDEXED,
        ACCESS_METHOD:=KEYED);
      RESETK(CONDMEANS,0);
      FINDK(CONDMEANS,0,COND.CONDITION);
    END;
  END;

```

```

        IF UFB(CONDMEANS)
            THEN
                WRITE(CONDMEANS,COND)
            ELSE
                WRITELN('entry in CONDMEANS.DAT already exists');
        CLOSE(CONDMEANS);
    END;

PROCEDURE READSEQFILE;
(* create matrix of DYNA sequences for attributes *)
VAR
    I,J,K:INTEGER;
    CH:CHAR;
BEGIN
    RESET(DYNASEQ);
    FOR I:=1 TO 30 DO
        FOR K:=1 TO 2 DO
            BEGIN
                FOR J:=1 TO 96 DO
                    BEGIN
                        READ(DYNASEQ,CH);
                        SEQS(I,J,K):=CH;
                    END;
                READLN(DYNASEQ);
            END;
        END;
    END;

PROCEDURE READNKCONS;
(* read file of constants for newman-keuls test *)
BEGIN
    RESET(DRTCONS);
    FOR I:=2 TO 12 DO READLN(DRTCONS,NKCONS[I]);
END;

PROCEDURE READWRITE(LINES:INTEGER);
(* read lines from drtres.dat and output to printer *)
VAR
    I:INTEGER;
    CH:CHAR;
BEGIN
    FOR I:=1 TO LINES DO
        BEGIN
            WHILE NOT EOLN(DRTRES) DO
                BEGIN
                    READ(DRTRES,CH);
                    WRITE(CH);
                END;
            WRITELN;
            READLN(DRTRES);
        END;
    END;

PROCEDURE READRESFIL;
(* read drtres and build up matrices of results *)

```

```

VAR
  CH,VERS,AORB:CHAR;
  RIGHTS,WRONGS,L,OPLINES:INTEGER;
  NUMTLK,TOTPERATT:INTEGER;
  ATT,SEQ:INTEGER;
BEGIN
  FOR I:=1 TO 3 DO WRITELN;
  RESET(DRTRES);
  FOR I:=1 TO 14 DO BEGIN READ(DRTRES,CH); WRITE(CH);END;
  READLN(DRTRES,EXPID);
  WRITELN(EXPID);
  READLN(DRTRES);
  WRITELN;
  (* output experiment details *)
  FOR I:=1 TO 31 DO BEGIN READ(DRTRES,CH); WRITE(CH);END;
  READLN(DRTRES,VERS);
  WRITELN(VERS);
  IF VERS='1' THEN BEGIN
    READWRITE(21);
    FOR I:=1 TO 15 DO
      BEGIN
        READ(DRTRES,CH);
        WRITE(CH);
      END;
    END
  ELSE
    BEGIN
      READWRITE(18);
      FOR I:=1 TO 6 DO
        BEGIN
          READ(DRTRES,CH);
          WRITE(CH);
        END;
      END;
    END;
  READLN(DRTRES,NUMOFPRES);
  WRITELN(NUMOFPRES:3);
  READWRITE(1);
  (* read list numbers and calculate which row
    in matrix SEQ each presentation corresponds to *)
  NUMTLK:=0; SEQNUM[1]:=1000;(* tested later for attributes *)
  FOR I:=1 TO NUMOFPRES DO
    BEGIN
      IF I MOD 2 <> 0
      THEN (* build up talker id strings *)
        BEGIN
          NUMTLK:=NUMTLK+1;
          FOR J:=1 TO 3 DO
            BEGIN
              READ(DRTRES,TLKIDARR[J]);
              WRITE(TLKIDARR[J])
            END;
          PACK(TLKIDARR,1,TLKID[NUMTLK]);
          FOR J:=1 TO 13 DO BEGIN READ(DRTRES,CH);WRITE(CH) END
        END
    END

```

```

ELSE
    FOR J:=1 TO 16 DO BEGIN READ(DRTRES,CH);WRITE(CH) END;
READ(DRTRES,CH);
WRITE(CH);
IF CH='.'
THEN
    BEGIN
        READ(DRTRES,CH);
        WRITE(CH)
    END
ELSE
    BEGIN
        SEQNUM[NUMTLK]:=ORD(CH)-48;
        READ(DRTRES,CH);
        WRITE(CH);
        SEQNUM[NUMTLK]:=10*SEQNUM[NUMTLK]+ORD(CH)-48;
        READ(DRTRES,AORB);
        IF AORB='B' THEN SEQNUM[NUMTLK]:=SEQNUM[NUMTLK]+15;
        WRITE(AORB);
        FOR J:=1 TO 2 DO BEGIN READ(DRTRES,CH);WRITE(CH);END;
        END;
    READLN(DRTRES);
    WRITELN;
    END;
READLN(DRTRES);
FOR I:=1 TO 10 DO BEGIN READ(DRTRES,CH); WRITE(CH);END;
READLN(DRTRES,NUMOFLISNERS);
WRITELN(NUMOFLISNERS:3);
READWRITE(1);
(* build up listener ID strings *)
FOR I:=1 TO NUMOFLISNERS DO
    BEGIN
        FOR J:=1 TO 4 DO
            BEGIN
                READ(DRTRES,LNRIDARR[J]);
                WRITE(LNRIDARR[J])
            END;
        PACK(LNRIDARR,1,LNRID[I]);
        WHILE NOT EOLN (DRTRES) DO BEGIN
            READ(DRTRES,CH);
            WRITE(CH);
            END;
        READLN(DRTRES);WRITELN;
        END;
    READLN(DRTRES);WRITELN;
    WHILE NOT EOLN (DRTRES) DO READ(DRTRES,CH);
    READLN(DRTRES);
    NUMOFTALKERS:=NUMOFFRES DIV 2;
    (* read 0 and 1's for each pres and build up results matrix *)
    FOR J:=1 TO NUMOFTALKERS DO
        FOR K:=1 TO NUMOFREPS DO
            FOR I:=1 TO NUMOFLISNERS DO
                BEGIN
                    FOR L:=1 TO 2 DO

```

```

        BEGIN
        WHILE NOT EOLN (DRTRES) DO READ(DRTRES,CH);
        READLN(DRTRES);
        END;
        RIGHTS:=0; WRONGS:=0;
        SEQJ:=0; ATT:=0;
        FOR L:=1 TO 3 DO
            BEGIN
            WHILE NOT EOLN (DRTRES) DO
            BEGIN
                SEQJ:=SEQJ+1;
                ATT:=ATT+1;
                IF ATT=7 THEN ATT:=1;
                READ(DRTRES,CH);
                IF CH='1' THEN BEGIN
                    RIGHTS:=RIGHTS+1;
                    IF SEQNUM[1]<>1000 THEN
                    IF SEQ[SEQNUM[J],SEQJ,K]='0'
                        THEN PORA[I,1,ATT]:=
                            PORA[I,1,ATT] + 1
                        ELSE PORA[I,2,ATT]:=
                            PORA[I,2,ATT] + 1;
                    END
                ELSE WRONGS:=WRONGS+1;
            END;
            READLN(DRTRES);
        END;
        RESMAT[I,J,K]:=((RIGHTS-WRONGS)*100)/
            (RIGHTS+WRONGS);
    END;

    (* apply conversion to attributes *)
    TOTPERATT:=NUMOFPRES * 8;
    FOR I:= 1 TO NUMOFLISNERS DO
        FOR J:=1 TO 2 DO
            FOR K:=1 TO 6 DO
                PORA[I,J,K]:=((2*PORA[I,J,K]-TOTPERATT)*100)
                    /TOTPERATT;
            END;
        END;
    END;

    PROCEDURE SINGLEREPS;
    (* average repetitions for tukey test *)
    BEGIN
        FOR I:=1 TO NUMOFLISNERS DO
            FOR J:=1 TO NUMOFTALKERS DO
                SINREPMAT[I,J]:=(RESMAT[I,J,1]+RESMAT[I,J,2])/2;
            END;
        END;
    END;

    PROCEDURE SETUNLNS;
    (* calculate underlining for NK test *)
    VAR STEPS:INTEGER;
    BEGIN
        STEPS:=1;
        FOR I:=1 TO 12 DO

```

```

        BEGIN
        UNLNS[I]:=STEPS;
        STEPS:=STEPS+2;
        END;
END;

PROCEDURE NKCALCS(TREATMEANS:ARROFMEANS;
                  TKLN:INTEGER);
(* newman-keuls test for listeners and talkers *)
LABEL
    99;
VAR
    CONV,DASHES,LINESUB,NUMTKLN,PREVIOUS,TEMP2:INTEGER;
    SIGDIFF,SORTED:BOOLEAN;
    TEMP,Q99MULT:REAL;
    NKMAT:ARRAY[1..12,1..13] OF REAL;
    LINELEN:ARRAY[1..40] OF INTEGER;
    POSITIONARR:ARRAY[1..12] OF INTEGER;
BEGIN
    READNKCONS;
    NUMTKLN:=TKLN;
    FOR I:=1 TO NUMTKLN DO POSITIONARR[I]:=I;
    CONV:=NUMOFTALKERS+NUMOFLISNERS-NUMTKLN;
    (* sort *)
    FOR I:=1 TO NUMTKLN DO
        BEGIN
            SORTED:=TRUE;
            FOR J:=1 TO (NUMTKLN-1) DO
                IF TREATMEANS[J]>TREATMEANS[J+1] THEN
                    BEGIN
                        TEMP:=TREATMEANS[J];
                        TREATMEANS[J]:=TREATMEANS[J+1];
                        TREATMEANS[J+1]:=TEMP;
                        TEMP2:=POSITIONARR[J];
                        POSITIONARR[J]:=POSITIONARR[J+1];
                        POSITIONARR[J+1]:=TEMP2;
                    END;
                    SORTED:=FALSE;
                END;
            END;
        END;
    (* create matrix *)
    FOR I:=1 TO NUMTKLN-1 DO
        FOR J:=1 TO NUMTKLN-I DO
            NKMAT[I,J+I]:=TREATMEANS[J+I]-TREATMEANS[I];
        END;
    (* calculate constant table multiplier *)
    Q99MULT:=SQRT(MSERR/(NUMOFREPS*CONV));
    (* ammend matrix to contain 999 for ** in Winer *)
    IF NKMAT[1,NUMTKLN]<(NKCONS[1]*Q99MULT)
    (* i.e. if no sigdiff in top right of matrix *)
    (* then print 1 - numtkln and underline *)
    THEN
        BEGIN
            WRITE(' ');
            FOR I:=1 TO NUMTKLN DO WRITE(POSITIONARR[I]:2);
            WRITELN;
        END;
    END;

```

```

WRITE(' ');
  FOR I:=1 TO NUMTKLN*2-1 DO WRITE('-');
END
ELSE
  BEGIN
    NKMAT[1,NUMTKLN]:=999;
    FOR I:=2 TO NUMTKLN-1 DO
      FOR J:=0 TO I-1 DO
        IF NKMAT[I-J,NUMTKLN-J] >
          (NKCONS[NUMTKLN+1-I]*Q99MULT)
        THEN NKMAT[I-J,NUMTKLN-J]:=999;
    (* set up underlines *)
    FOR I:=1 TO NUMTKLN-1 DO
      BEGIN
        LINELEN[I]:=1;
        FOR J:=I+1 TO NUMTKLN DO
          BEGIN
            (* count non-sigdiffs *)
            IF NKMAT[I,J] <> 999
            THEN LINELEN[I]:=LINELEN[I]+1
            ELSE GOTO 99;
          END;
        99:      END;
      (* printout *)
      WRITELN;
      WRITE(' ');
      FOR I:=1 TO NUMTKLN DO WRITE(POSITIONARR[I]:2);
      WRITELN;
      SETUNLNS;
      DASHES:=1;
      PREVIOUS:=1;
      FOR I:=1 TO NUMTKLN-1 DO
        BEGIN
          IF LINELEN[I] <> 1 THEN
            BEGIN
              FOR J:=1 TO DASHES DO WRITE(' ');
              IF (2*DASHES)+UNLNS[LINELEN[I]]<>PREVIOUS THEN
                FOR J:=1 TO UNLNS[LINELEN[I]] DO WRITE('-');
              WRITELN;
              END;
              PREVIOUS:=(2*DASHES)+UNLNS[LINELEN[I]];
              DASHES:=DASHES+1;
            END;
          END;
        END;
      END;
    END;
  END;
PROCEDURE CALCLMEANS;
VAR
  MEANSTOT:REAL;
BEGIN
  (* calc means for N.K. *)
  FOR I:=1 TO NUMOFLISNERS DO
    BEGIN
      MEANSTOT:=0;
      FOR J:=1 TO NUMOFTALKERS DO

```

```

        MEANSTOT:=MEANSTOT+SINREPMAT[I,J];
    LMEANS[I]:=MEANSTOT/NUMOFTALKERS;
    END;
END;
PROCEDURE CALCTMEANS;
VAR
    MEANSTOT:REAL;
BEGIN
    (* calc tmeans for N.K. *)
    FOR J:=1 TO NUMOFTALKERS DO
        BEGIN
            MEANSTOT:=0;
            FOR I:=1 TO NUMOFLISNERS DO
                MEANSTOT:=MEANSTOT+SINREPMAT[I,J];
            TMEANS[J]:=MEANSTOT/NUMOFLISNERS;
        END;
    END;
END;
PROCEDURE STATSONATTS;
BEGIN
    (* stats on attributes *)
    FOR J:=1 TO 2 DO
        FOR K:=1 TO 6 DO
            BEGIN
                ATTMEAN[J,K]:=0;
                ATTSS[J,K]:=0;
                MATTSS[K]:=0
            END;
        FOR J:=1 TO 2 DO
            FOR K:=1 TO 6 DO
                BEGIN
                    FOR I:=1 TO NUMOFLISNERS DO
                        BEGIN
                            ATTMEAN[J,K]:=ATTMEAN[J,K]+PORA[I,J,K];
                            ATTSS[J,K]:=ATTSS[J,K]+SQR(PORA[I,J,K]);
                        END;
                    ATTMEAN[J,K]:=ATTMEAN[J,K]/NUMOFLISNERS;
                END;
            (* mean of P or A *)
            FOR K:=1 TO 6 DO
                PAMEAN[K]:=(ATTMEAN[1,K]+ATTMEAN[2,K])/2;
            (* SS of P or A *)
            FOR K:=1 TO 6 DO
                FOR I:=1 TO NUMOFLISNERS DO
                    FOR J:=1 TO 2 DO
                        MATTSS[K]:=MATTSS[K]+SQR(PORA[I,J,K]);
                    (* SE of P or A *)
                    FOR I:=1 TO 2 DO
                        FOR J:=1 TO 6 DO
                            IF NUMOFLISNERS=1
                                THEN
                                    ATTSE[I,J]:=0
                                ELSE
                                    ATTSE[I,J]:=SQRT((ATTSS[I,J]-NUMOFLISNERS*SQR(ATTMEAN[I,J]))
                                        )/(NUMOFLISNERS*(NUMOFLISNERS-1)));

```



```

(* SE of means of P and A *)
NVAL:=NUMOFLISNERS*2;
IF NUMOFLISNERS>1 THEN
FOR J:=1 TO 6 DO
    ATTSE[3,J]:=SQRT((MATTSS[J]-NVAL*SQR(PAMEAN[J]))
                    /(NVAL*(NVAL-1)));

(* printout *)
TITLSTR:='Voicing      Nasality      Sustention Sibilation Graveness
Compactness';
TITLPTR:=1;
FOR I:=1 TO 4 DO WRITELN;
WRITELN('      ATTRIBUTE SCORES');
WRITELN('      -----');
WRITELN; WRITELN;
WRITELN('              Present      Absent      Mean');
WRITELN('              mean  SE      mean  SE      mean  SE');
WRITELN;
    FOR I:=1 TO 6 DO
        BEGIN
            SUBTITL:=SUBSTR(TITLSTR,TITLPTR,11);
            WRITELN(SUBTITL,' ',ATTMEAN[1,I]:8:1,ATTSE[1,I]:6:1,
                    ATTMEAN[2,I]:8:1,ATTSE[2,I]:6:1,PAMEAN[I]:8:1,
                    ATTSE[3,I]:6:1);
            TITLPTR:=TITLPTR+11;
        END;
    END;

PROCEDURE ANOVAR;
(* main statistics calculation and output routine *)
VAR
    G,TOT3,TOT4,TOT5,TOT6,TOT7,TOT8,TOT9:REAL;
    TOT3SUB1,TOT4SUB1,TOT6SUB,TOT7SUB:REAL;
    GRANDTOT,SDTOT:REAL;
    NONADDQUOT,NONADDSUB1,NONADDSUB2,NONADDIV:REAL;
    SSNONADD,SSBAL,MSBAL,SSRES:REAL;
    MSROW,ROWRAT,MSCOL,COLRAT,MSROWCOL,ROWCOLRAT:REAL;
    LISNDF,TALKDF,LNTKDF,ERRDF,TOTDF,BALDF:INTEGER;
    COMPNT:ARRAY[1..12] OF REAL;
    COMPNT10:ARRAY[1..12] OF REAL;
    TSSROW,TSSCOL,TMSROW,TMSCOL:REAL;
    COMPNT12:ARRAY[1..12] OF REAL;
    SDLN,SDTK:ARRAY[1..12] OF REAL;
    BEGIN
        G:=0;COMPNT[2]:=0;TOT3:=0;TOT4:=0;TOT5:=0;
        FOR I:=1 TO NUMOFLISNERS DO
            BEGIN
                TOT3SUB1:=0;
                FOR J:=1 TO NUMOFTALKERS DO
                    BEGIN
                        TOT3SUB1:=TOT3SUB1
                            +RESMAT[I,J,1]+RESMAT[I,J,2];
                        TOT5:=TOT5+
                            (RESMAT[I,J,1]+RESMAT[I,J,2])**2;
                        FOR K:=1 TO NUMOFREPS DO
                            BEGIN

```

```

        G:=G+RESMAT[I,J,K];
        COMPNT[2]:=COMPNT[2]+
            RESMAT[I,J,K]**2;
    END;
END;
TOT3:=TOT3+TOT3SUB1**2;
END;
(* build components required in specification *)
COMPNT[1]:=G**2/(NUMOFLISNERS*NUMOFTALKERS
    *NUMOFREPS);
COMPNT[3]:=TOT3/(NUMOFTALKERS*NUMOFREPS);
COMPNT[5]:=TOT5/NUMOFREPS;
TOT4:=0;
FOR J:=1 TO NUMOFTALKERS DO
    BEGIN
        TOT4SUB1:=0;
        FOR I:=1 TO NUMOFLISNERS DO
            BEGIN
                TOT4SUB1:=TOT4SUB1+RESMAT[I,J,1]
                    +RESMAT[I,J,2];
            END;
        TOT4:=TOT4+TOT4SUB1**2;
    END;
COMPNT[4]:=TOT4/(NUMOFLISNERS*NUMOFREPS);
    SSROW:=COMPNT[3]-COMPNT[1];
    SSCOL:=COMPNT[4]-COMPNT[1];
    SSROWCOL:=COMPNT[5]-COMPNT[3]-COMPNT[4]+COMPNT[1];
    SSERR:=COMPNT[2]-COMPNT[5];
    SSTOTAL:=COMPNT[2]-COMPNT[1];
    LISNDF:=NUMOFLISNERS-1;
    TALKDF:=NUMOFTALKERS-1;
    LNTKDF:=LISNDF*TALKDF;
    ERRDF:=NUMOFLISNERS*NUMOFTALKERS*(NUMOFREPS-1);
    TOTDF:=NUMOFLISNERS*NUMOFTALKERS*NUMOFREPS-1;
    MSROW:=SSROW/LISNDF;
    MSCOL:=SSCOL/TALKDF;
    MSROWCOL:=SSROWCOL/LNTKDF;
    MSERR:=SSERR/ERRDF;
    ROWRAT:=MSROW/MSERR;
    COLRAT:=MSCOL/MSERR;
    ROWCOLRAT:=MSROWCOL/MSERR;
(* Tukey Test for Nonadditivity *)
SINGLEREPS;
TOT6:=0; TOT8:=0; TOT9:=0;
FOR I:=1 TO NUMOFLISNERS DO
    BEGIN
        TOT6SUB:=0;
        FOR J:=1 TO NUMOFTALKERS DO
            BEGIN
                TOT6SUB:=TOT6SUB+SINREPMAT[I,J];
                TOT8:=TOT8+SINREPMAT[I,J];
                TOT9:=TOT9+SINREPMAT[I,J]**2;
            END;
        TOT6:=TOT6+TOT6SUB**2;
    
```

```

        COMPNT10[I]:=TOT6SUB/NUMOFTALKERS;
        END;
COMPNT[6]:=TOT6/NUMOFTALKERS;
COMPNT[8]:=TOT8**2/(NUMOFLISNERS*NUMOFTALKERS);
COMPNT[9]:=TOT9;
COMPNT[11]:=TOT8/(NUMOFLISNERS*NUMOFTALKERS);
TOT7:=0;
FOR J:=1 TO NUMOFTALKERS DO
    BEGIN
        TOT7SUB:=0;
        FOR I:=1 TO NUMOFLISNERS DO
            BEGIN
                TOT7SUB:=TOT7SUB+SINREPMAT[I,J];
            END;
        TOT7:=TOT7+TOT7SUB**2;
        COMPNT12[J]:=TOT7SUB/NUMOFLISNERS;
    END;
COMPNT[7]:=TOT7/NUMOFLISNERS;
TSSROW:=COMPNT[6]-COMPNT[8];
TSSCOL:=COMPNT[7]-COMPNT[8];
SSRES:=COMPNT[9]-COMPNT[6]-COMPNT[7]+COMPNT[8];
(* Calc Nonadditivity *)
NONADDQUOT:=0; NONADDSUB2:=0;
FOR I:=1 TO NUMOFLISNERS DO
    BEGIN
        NONADDSUB1:=0 ;
        FOR J:=1 TO NUMOFTALKERS DO
            BEGIN
                NONADDSUB1:=NONADDSUB1+(COMPNT12[J]-COMPNT[11])
                    *SINREPMAT[I,J];
            END;
        NONADDSUB2:=NONADDSUB2+(COMPNT10[I]-COMPNT[11])**2;
        NONADDQUOT:=NONADDQUOT+NONADDSUB1
            *(COMPNT10[I]-COMPNT[11]);
    END;
NONADDQUOT:=NONADDQUOT**2;
NONADDIV:=0;
FOR J:=1 TO NUMOFTALKERS DO
    BEGIN
        NONADDIV:=NONADDIV+(COMPNT12[J]-COMPNT[11])**2;
    END;
IF NONADDIV = 0 THEN NONADDIV:= 0.001;
NONADDIV:=NONADDSUB2*NONADDIV;
SSNONADD:=NONADDQUOT/NONADDIV;
SSBAL:=SSRES-SSNONADD;
BALDF:=LNTKDF-1;
TMSROW:=TSSROW/LISNDF;
TMSCOL:=TSSCOL/TALKDF;
MSBAL:=SSBAL/BALDF;
FOR I:=1 TO 12 DO WRITELN(' ');
WRITELN('          TUKEY TEST FOR NONADDATIVITY');
WRITELN('          -----');
FOR I:=1 TO 3 DO WRITELN;
WRITELN('Source          Sum of          Degrees of          Mean ');

```

```

WRITELN('          Squares    Freedom    Square');
WRITELN(' ');
WRITELN('LISNERS ',TSSROW:13:2,LISNDF:10,TMSROW:13:2);
WRITELN('TALKERS ',TSSCOL:13:2,TALKDF:10,TMSCOL:13:2);
WRITELN('RESIDUAL',SSRES:13:2,LNTKDF:10);
WRITELN('NONADD ',SSNONADD:13:2,'          1',SSNONADD:13:2);
WRITELN('BALANCE ',SSBAL:13:2,BALDF:10,MSBAL:13:2);
FOR I:=1 TO 5 DO WRITELN;
WRITELN('      ANALYSIS OF VARIANCE');
WRITELN('      -----');
WRITELN(' ');WRITELN(' ');
WRITELN('Source          Sum of    Degrees of    Mean',
      '          F-Ratio');

WRITELN('          Squares    Freedom    Square');
WRITELN(' ');
WRITELN('LISNERS ',SSROW:13:2,LISNDF:10,MSROW:13:2,ROWRAT:11:2);
WRITELN('TALKERS ',SSCOL:13:2,TALKDF:10,MSCOL:13:2,COLRAT:11:2);
WRITELN('LN * TK ',SSROWCOL:13:2,LNTKDF:10,MSROWCOL:13:2,
      ROWCOLRAT:11:2);

WRITELN('ERROR ',SSERR:13:2,ERRDF:10,MSERR:13:2);
WRITELN('TOTAL ',SSTOTAL:13:2,TOTDF:10);
(* Newman - Keuls Test *)
FOR I:=1 TO 4 DO WRITELN;
WRITELN('      NEWMAN - KEULS TEST');
WRITELN('      -----');WRITELN;
WRITELN('LISTENERS :');WRITELN;
NKCALCS(LMEANS,NUMOFLISNERS);
WRITELN;WRITELN;WRITELN('TALKERS :');
NKCALCS(TMEANS,NUMOFTALKERS);
IF SEQNUM[1]<>1000 THEN STATSONATTS;
(* calculate treatment means for basic stats *)
SDTOT:=0;GRANDTOT:=0;
FOR I:=1 TO NUMOFLISNERS DO
  BEGIN
    SDLN[I]:=0;
    FOR J:=1 TO NUMOFTALKERS DO
      SDLN[I]:=SDLN[I]+SINREPMAT[I,J]**2;
    SDLN[I]:=SQRT(ABS(SDLN[I]-NUMOFTALKERS*(LMEANS[I])**2)
      /(NUMOFTALKERS-1));
    SDTOT:=SDTOT+(LMEANS[I])**2;
    GRANDTOT:=GRANDTOT+LMEANS[I];
  END;
GRANDMEAN:=GRANDTOT/NUMOFLISNERS;
STANDEV:=SQRT(ABS(SDTOT-NUMOFLISNERS*GRANDMEAN**2)
  /(NUMOFLISNERS-1));
OASTANDEV:=STANDEV;
STANERR:=STANDEV/SQRT(NUMOFLISNERS);
FOR I:=1 TO 10 DO WRITELN;
WRITELN('      INDIVIDUAL SCORES');
WRITELN('      -----');
FOR I:=1 TO 3 DO WRITELN;
WRITE('LNR  T1');
FOR I:=2 TO NUMOFTALKERS DO WRITE('  T',I:1);

```

```

WRITE('  SD');
Writeln;
FOR I:=1 TO NUMOFLISNERS DO
  BEGIN
    WRITE(I:2);
    FOR J:=1 TO NUMOFTALKERS DO WRITE(SINREPMAT[I,J]:6:1);
    WRITE(SDLN[I]:6:1);
    Writeln;
  END;
Writeln;
Writeln('LNR  MEAN OVER TKR');
FOR I:=1 TO NUMOFLISNERS DO
  Writeln(I:2,LMEANS[I]:13:1);
Writeln;
Writeln('STD. DEVN. =',STANDEV:5:1);
Writeln('STD. ERROR =',STANERR:5:1);
OASTANERR:=STANERR;
SDTOT:=0;
FOR J:=1 TO NUMOFTALKERS DO
  BEGIN
    SDTK[J]:=0;
    FOR I:= 1 TO NUMOFLISNERS DO
      SDTK[J]:=SDTK[J]+SINREPMAT[I,J]**2;
    SDTK[J]:=SQRT((SDTK[J]-NUMOFLISNERS*TMEANS[J]**2)
                  /(NUMOFLISNERS-1));
    SDTOT:=SDTOT+(TMEANS[J])**2;
  END;
STANDEV:=SQRT((SDTOT-NUMOFTALKERS*GRANDMEAN**2)
              /(NUMOFTALKERS-1));
STANERR:=STANDEV/SQRT(NUMOFTALKERS);
FOR I:=1 TO 5 DO Writeln;
WRITE('TKR  L1');
FOR I:=2 TO NUMOFLISNERS DO
  IF I<10 THEN WRITE('  L',I:1) ELSE WRITE('  L',I:1);
WRITE('  SD');
Writeln;
FOR I:=1 TO NUMOFTALKERS DO
  BEGIN
    WRITE(I:2);
    FOR J:=1 TO NUMOFLISNERS DO WRITE(SINREPMAT[J,I]:6:1);
    WRITE(SDTK[I]:6:1);
    Writeln;
  END;
Writeln;
Writeln('TKR  MEAN OVER LNR');
FOR I:=1 TO NUMOFTALKERS DO
  Writeln(I:2,TMEANS[I]:13:1);
Writeln;
Writeln('STD. DEVN. =',STANDEV:5:1);
Writeln('STD. ERROR =',STANERR:5:1);
FOR I:=1 TO 5 DO Writeln;
Writeln('  DRT SCORE =',GRANDMEAN:5:1);
Writeln('STANDARD ERROR =',OASTANERR:5:1);
FOR I:=1 TO 5 DO Writeln;

```

```

END;

PROCEDURE BASIC_AND_ATT_STATS;

BEGIN
  IF SEQNUM[1] <>1000 THEN STATSONATTS;
  FOR I:= 1 TO 10 DO WRITELN;
  WRITELN('      INDIVIDUAL SCORES');
  WRITELN('      -----');
  FOR I:=1 TO 3 DO WRITELN;
  WRITE('LNR');
  FOR I:=1 TO NUMOFTALKERS DO WRITE('      T',I:1);
  WRITELN;
  FOR I:=1 TO NUMOFLISNERS DO
    BEGIN
      WRITE(I:2,' ');
      FOR J:=1 TO NUMOFTALKERS DO WRITE(SINREPMAT[I,J]:6:1);
      WRITELN;
    END;
  WRITELN;
  WRITELN('LNR      MEAN OVER TKR');
  FOR I:=1 TO NUMOFLISNERS DO
    WRITELN(I:2,LMEANS[I]:13:1);
  WRITELN;
  FOR I:=1 TO 5 DO WRITELN;
  WRITE('TKR');
  FOR I:=1 TO NUMOFLISNERS DO
    WRITE('      L',I:1);
  WRITELN;
  FOR I:=1 TO NUMOFTALKERS DO
    BEGIN
      WRITE(I:2,' ');
      FOR J:=1 TO NUMOFLISNERS DO WRITE(SINREPMAT[J,I]:6:1);
      WRITELN;
    END;
  WRITELN;
  WRITELN('TKR      MEAN OVER LNR');
  FOR I:=1 TO NUMOFTALKERS DO
    WRITELN(I:2,TMEANS[I]:13:1);
  WRITELN;
  GRANDMEAN:=0;
  FOR J:=1 TO NUMOFTALKERS DO
    GRANDMEAN:=GRANDMEAN + TMEANS[J];
  GRANDMEAN:=GRANDMEAN/NUMOFTALKERS;
  FOR I:=1 TO 5 DO WRITELN;
  WRITELN('DRT SCORE = ',GRANDMEAN:5:1);
  FOR I:=1 TO 5 DO WRITELN;
  END;

  (* Main Program *)
  BEGIN
    WRITELN;
    FOR I:=1 TO 3 DO WRITELN;

```

```
INITIALISE;  
READSEQFILE;  
READRESFIL;  
SINGLEREPS;  
CALCLMEANS;  
CALCTMEANS;  
IF (NUMOFTALKERS > 2) AND  
  (NUMOFLISNERS > 2)  
  THEN ANOVAR  
ELSE BASIC_AND_ATT_STATS;  
UPDATECONDMEANS;  
END.
```

```

PROGRAM DRTFILE(INPUT,OUTPUT,AFL,BFL,DRTTMPFL1,DRTEDA,CONDMEANS);

(* Input:      Data from the file Condmeans created *)
(*           by DRTSTAT.                               *)
(* Processing: Extract required results, transpose *)
(*           into chars, reformat and store for      *)
(*           subsequent use by PANOVUN.                *)

CONST
    CREWNAMES=100;
    DEPTHOFNESTING=15;
    MAXDATASTRG=100;
    MAXTITLESTRG=100;

TYPE
    ARROFMEANS=ARRAY[1..12] OF REAL;
    STROFTWO=PACKED ARRAY[1..2] OF CHAR;
    CONDTYP=PACKED ARRAY[1..4] OF CHAR;
    ATTYP=PACKED ARRAY[1..3] OF CHAR;
    REALCHRS=PACKED ARRAY[1..5] OF CHAR;
    NAMTYP=PACKED ARRAY[1..6] OF CHAR;
    STROFTHREE=PACKED ARRAY[1..3] OF CHAR;
    CONDREC=RECORD
        CONDITION:[KEY(0)] PACKED ARRAY[1..7] OF CHAR;
        LNAMS:ARRAY[1..12] OF CONDTYP;
        LISNEROAMS:ARROFMEANS;
        ATTRIBUTES:ARRAY[1..12,1..2,1..6] OF REAL;
        TLKRESMAT:ARRAY[1..12,1..12] OF REAL;
        TNAMS:ARRAY[1..12] OF STROFTHREE;
        TLKNUM:INTEGER;
    END;

VAR
    CONDMEANS:FILE OF CONDREC;
    AFL,BFL,DRTTMPFL1,DRTEDA:TEXT;
    CONDLN,ATTLEN,LNUM,I,NUM:INTEGER;
    CONDNUMSTRG,ATTSTRG:VARYING[1000] OF CHAR;
    COND:CONDREC;
    PAM:ARRAY[1..12,1..3,1..6] OF REAL;
    CNUM:CONDTYP;
    CH:CHAR;
    FLSNOTFOUND:INTEGER;
    NAMES,PERSONS:ARRAY[1..CREWNAMES] OF NAMTYP;
    CREW,COL,EDACOLS:INTEGER;
    NESTING,NESTMAX:ARRAY[1..DEPTHOFNESTING] OF INTEGER;
    LISNERMISSING:ARRAY[1..CREWNAMES] OF INTEGER;
    INDEX,LIDNUM:ARRAY[1..CREWNAMES] OF STROFTWO;
    REQUDLNRS:ARRAY[1..CREWNAMES] OF NAMTYP;
    EXPECTED_VALS_PER_LNR:INTEGER;
    PERCENT:REAL;

FUNCTION INTOCHR(INTIN:INTEGER):STROFTWO;
(* turns integers(<99) into chars *)
VAR

```



```

    ICHRS:ARRAY[1..2] OF CHAR;
    ILOCSTRG:STROFTWO;
BEGIN
    ICHRS[1]:=CHR(48+(INTIN DIV 10));
    ICHRS[2]:=CHR(48+(INTIN REM 10));
    IF ICHRS[1]='0' THEN ICHRS[1]=' ';
    PACK(ICHRS,1,ILOCSTRG);
    INTTOCHR:=ILOCSTRG;
END;

FUNCTION CHRTOINT(CHIN:STROFTWO):INTEGER;
(* turns 2 char strings into integers *)
VAR
    INTOUT:INTEGER;
    CHRINTS:ARRAY[1..2] OF CHAR;
BEGIN
    UNPACK(CHIN,CHRINTS,1);
    IF CHRINTS[2]=' '
    THEN
        BEGIN CHRINTS[2]:=CHRINTS[1];CHRINTS[1]:='0' END;
    INTOUT:=(10*(ORD(CHRINTS[1])-48)) + (ORD(CHRINTS[2])-48);
    CHRTOINT:=INTOUT
END;

FUNCTION TOCHARS(LSNR:INTEGER;R:REAL):REALCHRS;
(* turns pos or neg real numbers i to chars of the form SAA.A *)
VAR
    IGER:INTEGER;
    CHRS:ARRAY[1..5] OF CHAR;
    LOCALSTR:REALCHRS;
BEGIN
    R:=(ROUND(R*10))/10;
    IF R=100 THEN TOCHARS:=' 100 '
    ELSE
        IF (R=0) AND (COND.LNAMS[LSNR]='----')
        THEN TOCHARS:=' -999'
        ELSE
            BEGIN
                IF R<0 THEN
                    BEGIN
                        CHRS[1]:='-';
                        R:=R*(-1)
                    END
                ELSE CHRS[1]:=' ';
                IGER:=TRUNC(R*10);
                CHRS[2]:=CHR(48+(IGER DIV 100));
                CHRS[3]:=CHR(48+(IGER DIV 10) REM 10);
                CHRS[4]:='.';
                CHRS[5]:=CHR(48+(IGER REM 10));
                IF CHRS[2]='0' THEN CHRS[2]=' ';
                IF (CHRS[1]='-') AND (CHRS[2]=' ')
                THEN
                    BEGIN
                        CHRS[1]:=' ';

```

```

        CHRS[2]:='- '
        END;
        PACK(CHRS,1,LOCALSTR);
        TOCHARS:=LOCALSTR
        END;

END;

PROCEDURE ENTERNAMES;
(* read listener names at start of each condition *)

VAR
    I,TIMESTHRU:INTEGER;
    MATCHED:BOOLEAN;

BEGIN
    FOR I:=1 TO 12 DO
        BEGIN
            TIMESTHRU:=1;
            READ(DRTTMPFL1,NAMES[I]);
            MATCHED:=FALSE;
            IF NAMES[I]<>' ----'
            THEN
                BEGIN
                    WHILE (NOT MATCHED) AND (TIMESTHRU<=CREW) DO
                        IF NAMES[I] = PERSONS(TIMESTHRU)
                        THEN
                            BEGIN
                                MATCHED:=TRUE;
                                INDEX[I]:=LIDNUM(TIMESTHRU)
                            END
                        ELSE
                            TIMESTHRU:=TIMESTHRU+1;
                    IF NOT MATCHED
                    THEN WRITELN('ERROR,   Unrecognised   listener   name:
',NAMES[I]);
                END;
            END;
            READLN(DRTTMPFL1);
        END;
    END;

PROCEDURE UPDATE(VAR CURRENT,MAX : INTEGER);
BEGIN
    IF CURRENT<MAX
    THEN
        CURRENT:=CURRENT+1
    ELSE
        BEGIN
            CURRENT:=1;
            COL:=COL+1;
            UPDATE(NESTING[COL],NESTMAX[COL]);
        END;
    END;
END;

```

PROCEDURE OUTPTNESTING;

BEGIN

```
    COL:=1;
    UPDATE(NESTING[COL],NESTMAX[COL]);
    FOR I:=1 TO EDACOLS+1 DO WRITE(BFL,' ',INTTOCHR(NESTING[I]));
END;
```

PROCEDURE TOEDAFORM;

(* reformat temp file to EDA format *)

VAR

```
    X,Y,I,J,K,L,CONDSETS,NUM,REQLNR,TEMP1:INTEGER;
    DATANAMES:VARYING[MAXDATASTRG] OF CHAR;
    TITLE:VARYING[MAXTITLESTRG] OF CHAR;
    DRT:NAMTYP;
    SCORE:ARRAY[1..12] OF NAMTYP;
    WHOBY:ARRAY[1..12] OF INTEGER;
    CH:CHAR;
    OK:BOOLEAN;
    REQLNRNUMS:ARRAY[1..12] OF STROFTWO;
    TEMP2:NAMTYP;
```

BEGIN

```
    (* read control file *)
    FOR I:=1 TO 10 DO READ(AFL,CH);
    READLN(AFL,REQLNR);
    FOR I:=1 TO REQLNR DO
        BEGIN
            READLN(AFL,REQUDLNRS[I]);
            LISNERMISSING[I]:=0;
            WHOBY[I]:=0;
        END;
    READLN(AFL,CH);
    IF CH <> '*'
        THEN WRITELN ('ERROR ** Caused by incorrect number of listener
IDs');
    READLN(AFL);
    READLN(AFL,EDACOLS);
    READLN(AFL,DATANAMES);
    NESTMAX[1] := REQLNR;
    FOR I:=2 TO EDACOLS+1 DO
        READ(AFL,NESTMAX[I]);
    READLN(AFL);
    READLN(AFL,TITLE);
    READLN(AFL,CH);
    IF CH<> '*' THEN WRITELN('ERROR ** Caused by too many format
lines');
    REPEAT
        READLN(AFL,CH)
    UNTIL CH='*';
    FOR I:=1 TO 16 DO READ(AFL,CH);
    READLN(AFL,CREW);
```

```

FOR I:=1 TO CREW DO
  READLN(AFL,PERSONS[I],CH,LIDNUM[I]); (* read listener ids *)
FOR I:=1 TO REQLNR DO (* note requd listener
ids *)
  FOR J:=1 TO CREW DO IF REQUDLNRS[I]=PERSONS[J] THEN
    REQLNRNUMS[I]:=LIDNUM[J];

NESTING[1]:=0;
FOR I:=2 TO EDACOLS+1 DO NESTING[I]:=1;
WRITELN(BFL,INTTOCHR(EDACOLS+3)); (* write to opfile *)
WRITELN(BFL,':"DRT  "',"INDIV  ","POSIT  "','DATANAMES);
WRITELN(BFL,TITLE);
RESET(DRTTMPFL1);
CONDSETS:=1;
FOR I:=3 TO EDACOLS+1 DO
  CONDSETS:=CONDSETS*NESTMAX[I];
EXPECTED_VALS_PER_LNR:=NESTMAX[2]*CONDSETS;
FOR I:=1 TO CONDSETS DO
  BEGIN
  ENTERNAMES;
  FOR J:=1 TO NESTMAX[2] DO
    BEGIN
    NUM:=1;
    FOR K:=1 TO 12 DO WHOBY[K]:=0;
    FOR K:=1 TO 12 DO
      BEGIN
      READ(DRTTMPFL1,DRT);
      OK:=FALSE;
      FOR L:=1 TO REQLNR DO IF NAMES[K]=REQUDLNRS[L]
        THEN OK:=TRUE;

      IF OK
      THEN
        BEGIN
        (* build up req listeners for sorting *)
        SCORE[NUM]:=DRT;
        WHOBY[NUM]:=CHRTOINT(INDEX[K]);
        NUM:=NUM+1
        END;
      END;
    (* check each req listener has a score *)
    WHILE (NUM-1)<REQLNR DO
      FOR X:=1 TO REQLNR DO
        BEGIN
        OK:=FALSE;
        FOR Y:=1 TO (NUM-1) DO
          IF WHOBY[Y]=CHRTOINT(REQLNRNUMS[X])
            THEN OK:=TRUE;
        IF NOT OK
        THEN
          BEGIN
          WHOBY[NUM]:=CHRTOINT(REQLNRNUMS[X]);
          SCORE[NUM]:=' -999';
          LISNERMISSING[X]:=LISNERMISSING[X]+1;
          NUM:=NUM+1
          END;
        END;
      END;
    END;
  END;

```

```

        END;
    (* sort *)
    FOR K:=1 TO REQLNR-1 DO
        FOR L:=1 TO REQLNR-1 DO
            IF WHOBY[L]>WHOBY[L+1]
            THEN
                BEGIN
                    TEMP1:=WHOBY[L];
                    WHOBY[L]:=WHOBY[L+1];
                    WHOBY[L+1]:=TEMP1;
                    TEMP2:=SCORE[L];
                    SCORE[L]:=SCORE[L+1];
                    SCORE[L+1]:=TEMP2
                END;
            (* print *)
            FOR K:=1 TO REQLNR DO
                BEGIN
                    WRITE(BFL,SCORE[K], '      ',INTTOCHR(WHOBY[K]));
                    OUTPTNESTING;
                    WRITELN(BFL)
                END;
            READLN(DRTTMPFL1);
            END;
        READLN(DRTTMPFL1);
        END;
    FOR I:=1 TO EDACOLS+3 DO WRITE(BFL,'-1 ');
    WRITELN(BFL);
    WRITELN(BFL,'-1');
END;

PROCEDURE DOEDAINPT;
(* create input file for panovun *)

VAR
    I:INTEGER;
    CH:CHAR;

BEGIN
    REWRITE(DRTEDA);
    RESET(AFL);
    FOR I:=1 TO 5 DO
        BEGIN
            CH:=' ';
            WHILE CH<>'*' DO READ(AFL,CH);
            READLN(AFL)
        END;
    READLN(AFL);
    CH:=' ';
    WHILE CH<>'*' DO
        BEGIN
            WHILE NOT EOLN(AFL) DO
                BEGIN
                    READ(AFL,CH);
                    IF CH<>'*' THEN WRITE(DRTEDA,CH)

```

```

        END;
        READLN(AFL);
        WRITELN(DRTEDA);
        END;
    END;

```

```

PROCEDURE OUTDATA(ATT:ATTYP);
(* decode attribute required and output *)

```

```

VAR
    LOCALARR:ARRAY[1..3] OF CHAR;
    TLKPOS,TPOS,TABNO,I,J:INTEGER;
    PAORM:CHAR;
BEGIN
    UNPACK(ATT,LOCALARR,1);
    PAORM:=LOCALARR[3];
    TABNO:=ORD(LOCALARR[1])+ORD(LOCALARR[2]);
    IF PAORM='P' THEN J:=1
        ELSE IF PAORM='A' THEN J:=2
            ELSE J:=3;

    WITH COND DO
        CASE TABNO OF
            144: BEGIN
                FOR I:=1 TO 12 DO
                    WRITE(DRTTMPFL1,' ',TOCHARS(I,LISNEROAMS(I)));
                END;
            165: BEGIN
                FOR I:=1 TO 12 DO
                    WRITE(DRTTMPFL1,' ',TOCHARS(I,PAM[I,J,1]));
                END;
            143: BEGIN
                FOR I:=1 TO 12 DO
                    WRITE(DRTTMPFL1,' ',TOCHARS(I,PAM[I,J,2]));
                END;
            168: BEGIN
                FOR I:=1 TO 12 DO
                    WRITE(DRTTMPFL1,' ',TOCHARS(I,PAM[I,J,3]));
                END;
            156: BEGIN
                FOR I:=1 TO 12 DO
                    WRITE(DRTTMPFL1,' ',TOCHARS(I,PAM[I,J,4]));
                END;
            153: BEGIN
                FOR I:=1 TO 12 DO
                    WRITE(DRTTMPFL1,' ',TOCHARS(I,PAM[I,J,5]));
                END;
            146: BEGIN
                FOR I:=1 TO 12 DO
                    WRITE(DRTTMPFL1,' ',TOCHARS(I,PAM[I,J,6]));
                END;
            159: BEGIN
                FOR I:=1 TO TLKNUM DO
                    BEGIN

```

```

        FOR J:=1 TO 12 DO
            WRITE (DRTTMPFL1, '
',TOCHARS(J,TLKRESMAT{J,I}));
            WRITELN(DRTTMPFL1);
        END;
    END;

    END;
    IF TABNO <> 159 THEN WRITELN(DRTTMPFL1);
END;

PROCEDURE SELECTDATA(COND:CONDREC);
(* write heading for block of data and loop thru attributes *)
(* calling OUTDATA. *)

VAR
    ANUM:INTEGER;
    ATTREQ:ATTYP;
BEGIN
    FOR I:=1 TO 12 DO
        WRITE(DRTTMPFL1, ' ',COND.LNAMS[I]);
        WRITELN(DRTTMPFL1);
        ANUM:=4;
        REPEAT
            ATTREQ:=SUBSTR(ATTSTRG,ANUM-3,3);
            OUTDATA(ATTREQ);
            ANUM:=ANUM+4
        UNTIL ANUM > ATTLEN;
        WRITELN(DRTTMPFL1)
    END;

PROCEDURE RETRIEVEREC(CONDNUMB:CONDTYP);
(* locate required conditions using EXPID as key, generate *)
(* means from Present and Absent, and call SELECTDATA *)

VAR
    EXPID:PACKED ARRAY[1..7] OF CHAR;
    I,J,K:INTEGER;

BEGIN
    EXPID:='CTL'+CONDNUMB;
    COND.CONDITION:=EXPID;
    OPEN(CONDMEANS,
        HISTORY:=UNKNOWN,
        ORGANIZATION:=INDEXED,
        ACCESS_METHOD:=KEYED);
    RESETK(CONDMEANS,0);
    FINDK(CONDMEANS,0,COND.CONDITION);
    IF NOT UFB(CONDMEANS)
        THEN
            BEGIN
                READ(CONDMEANS,COND);
                FOR I:=1 TO 12 DO
                    FOR K:=1 TO 6 DO

```

```

        BEGIN
        FOR J:=1 TO 2 DO
            PAM[I,J,K]:=COND.ATTRIBUTES[I,J,K];
            PAM[I,3,K]:=(PAM[I,1,K]+PAM[I,2,K])/2;
        END;
        SELECTDATA(COND);
        END
        ELSE
        BEGIN
        WRITELN(EXPID,'** not found **');
        FLSNOTFOUND:=FLSNOTFOUND+1;
        WRITELN(DRTTMPFL1)
        END;
        CLOSE(CONDMEANS);

        END;

```

```

PROCEDURE DECODESTRGS;
(* decode string of conditions reqd, and loop thru calling RETRIEVEREC
*)
VAR
    NUM1,NUM2,I:INTEGER;
    DELIM:PACKED ARRAY[1..1] OF CHAR;
    CONDARR,LASTARR:ARRAY[1..4] OF CHAR;
    ATTARR:ARRAY[1..2] OF CHAR;
    CONVS,CONDNUM,LASTNUM:COND TYP;
BEGIN
    CONDLN:=LENGTH(CONDNUMSTRG);
    ATTLEN:=LENGTH(ATTSTRG);
    LNUM:=1;
    REPEAT
        CONDNUM:=SUBSTR(CONDNUMSTRG,LNUM,4);
        DELIM:=SUBSTR(CONDNUMSTRG,LNUM+4,1);
        IF DELIM='- '
        THEN
            BEGIN
                LASTNUM:=SUBSTR(CONDNUMSTRG,LNUM+5,4);
                UNPACK(CONDNUM,CONDARR,1);
                UNPACK(LASTNUM,LASTARR,1);

                NUM1:=1000+100*(ORD(CONDARR[2])-48)+10*(ORD(CONDARR[3])-48)
                    +ORD(CONDARR[4])-48;

                NUM2:=1000+100*(ORD(LASTARR[2])-48)+10*(ORD(LASTARR[3])-48)
                    +ORD(LASTARR[4])-48;

                FOR I:=NUM1 TO NUM2 DO
                    BEGIN
                        CONDARR[4]:=CHR(48+(I REM 10));
                        CONDARR[3]:=CHR(48+((I REM 100) DIV 10));
                        CONDARR[2]:=CHR(48+((I REM 1000) DIV 100));
                        PACK(CONDARR,1,CONDNUM);
                        RETRIEVEREC(CONDNUM);
                    END
                END
            END
        END
    UNTIL LNUM=CONDLN;
END

```



```

        END;
        LNUM:=LNUM+5;
    END
ELSE
    RETRIEVEREC(CONDNUM);
    LNUM:=LNUM+5
    UNTIL LNUM > CONDLEN;
END;

BEGIN
OPEN(AFL,'infile',
    OLD);
RESET(AFL);
OPEN(BFL,'outfile');
REWRITE(BFL);
REWRITE(DRTTMPFL1);
CONDNUMSTRG:='';ATTSTRG:='';FLSNOTFOUND:=0;
FOR I:=1 TO 3 DO READLN(AFL);
READ(AFL,CH);
WHILE CH<>'*' DO
    BEGIN
        IF ORD(CH)=13 THEN CH:=' ';
        ATTSTRG:=ATTSTRG+CH;
        READ(AFL,CH);
    END;
READLN(AFL);READLN(AFL);
READ(AFL,CH);
WHILE CH<>'*' DO
    BEGIN
        IF ORD(CH)=13 THEN CH:=' ';
        CONDNUMSTRG:=CONDNUMSTRG+CH;
        READ(AFL,CH);
    END;
READLN(AFL);
FOR I:=1 TO 4 DO WRITELN;
WRITELN('OK, running DRTFILE');
WRITELN;
DECODESTRGS;
IF FLSNOTFOUND>0
THEN
    BEGIN
        WRITELN;
        WRITELN('Error!  **',FLSNOTFOUND:3,' files not found **');
        WRITELN;
    END
ELSE
    BEGIN
        WRITELN('OK, all records found, creating output file');
        WRITELN;
        TOEDAFORM;
        WRITELN('OK, output file created');
        WRITELN;
        FOR I:=1 TO CREW DO
            IF LISNERMISSING[I]>0

```

```

      THEN
      BEGIN
      PERCENT:=(LISNERMISSING[I] / EXPECTED_VALS_PER_LNR)*100;

      WRITELN('Warning ','REQUDLNRS[I],' has ',
      LISNERMISSING[I]:3,' (' ,PERCENT:3:1,'%') missing
values');
      END;
      DOEDAINPT;
      END;
      END.

```

PROGRAM DRTEDIT(INPUT,OUTPUT,CONDMEANS);

TYPE

COND TYP=PACKED ARRAY[1..4] OF CHAR;
CONDNAMETYP=PACKED ARRAY[1..7] OF CHAR;
STROFTHREE=PACKED ARRAY[1..3] OF CHAR;
ARROFMEANS=ARRAY[1..12] OF REAL;
CONDREC=RECORD
 CONDITION:[KEY(0)] PACKED ARRAY[1..7] OF CHAR;
 LNAMS:ARRAY[1..12] OF COND TYP;
 LISNEROAMS:ARROFMEANS;
 ATTRIBUTES:ARRAY[1..12,1..2,1..6] OF REAL;
 TLKRESMAT:ARRAY[1..12,1..12] OF REAL;
 TNAMS:ARRAY[1..12] OF STROFTHREE;
 TLKNUM:INTEGER;
END;

VAR

CONDMEANS:FILE OF CONDREC;
CHOICE:INTEGER;
COND:CONDREC;

PROCEDURE DISPINDEX;

VAR

 WRITS:INTEGER;
 NUM:COND TYP;
BEGIN
 WRITELN;
 RESET(CONDMEANS);
 WRITS:=-1;
 WHILE NOT EOF(CONDMEANS) DO
 BEGIN
 READ(CONDMEANS,COND);
 NUM:=SUBSTR(COND.CONDITION,4,4);
 WRITS:=WRITS+1;
 IF WRITS>12
 THEN
 BEGIN
 WRITELN;
 WRITS:=0
 END;
 WRITE(NUM,' ');
 END;
 WRITELN;
 CLOSE(CONDMEANS);
END;

PROCEDURE DELENTY;

VAR

 CNUM:COND TYP;
 EXPID:CONDNAMETYP;
BEGIN
 OPEN(CONDMEANS,
 HISTORY:=UNKNOWN,

```

        ORGANIZATION:=INDEXED,
        ACCESS_METHOD:=KEYED);
WRITE('Enter Condition Number to be Deleted: ');
READLN(CNUM);
EXPID:='CTL'+ CNUM;
FINDK(CONDMEANS,0,EXPID);
IF NOT UFB(CONDMEANS)
THEN
    BEGIN
        DELETE(CONDMEANS);
        WRITELN(EXPID,' deleted OK')
    END
ELSE
    WRITELN('Condition ',EXPID,' not on file');
CLOSE(CONDMEANS)
END;

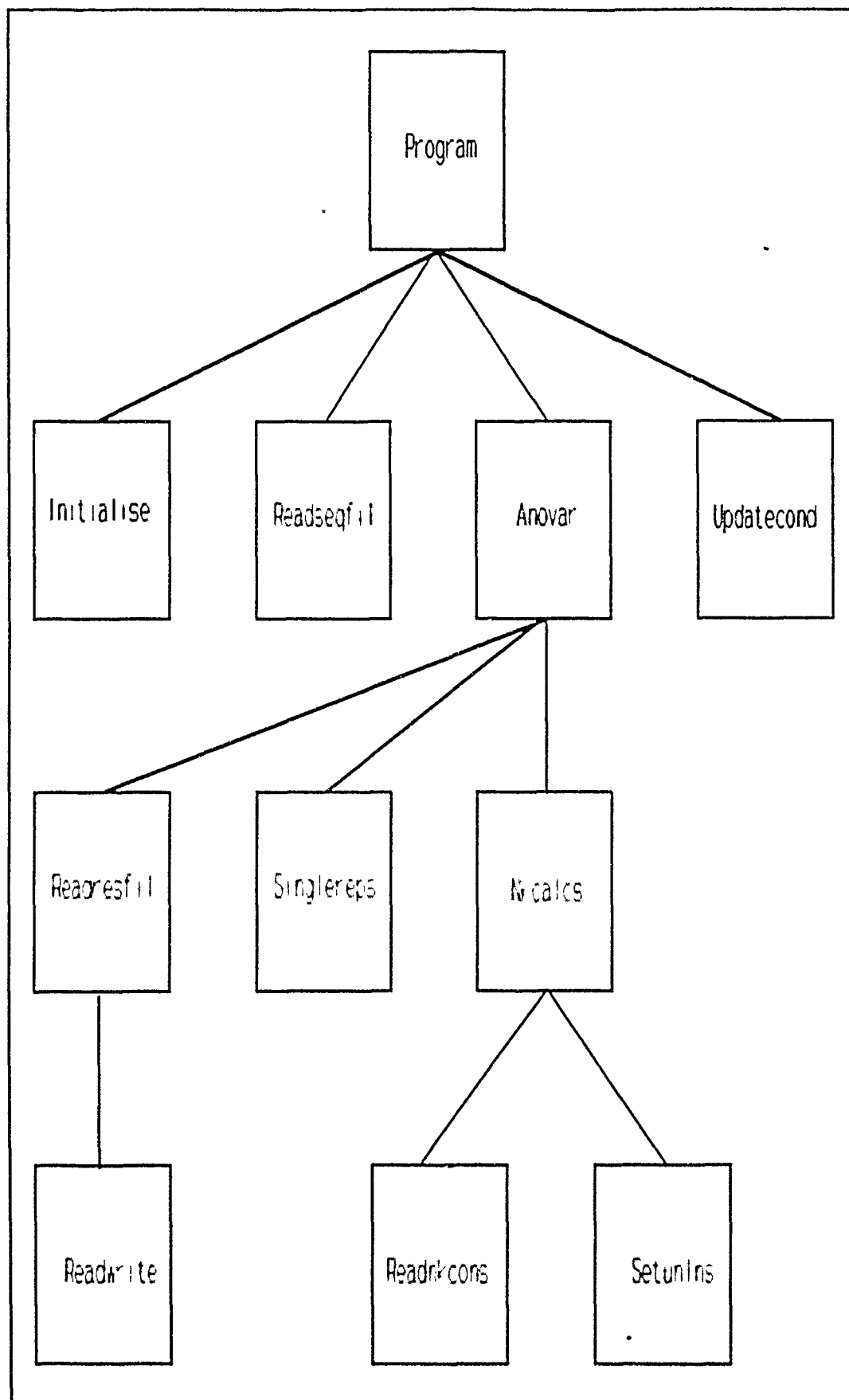
PROCEDURE DISPMENU;
    BEGIN
        WRITELN;
        WRITELN('DRTEDIT');
        WRITELN('-----');
        WRITELN;
        WRITELN('1. Display Index');
        WRITELN('2. Delete an entry');
        WRITELN('3. End');
        WRITELN;
        WRITE('Select.. ');
    END;

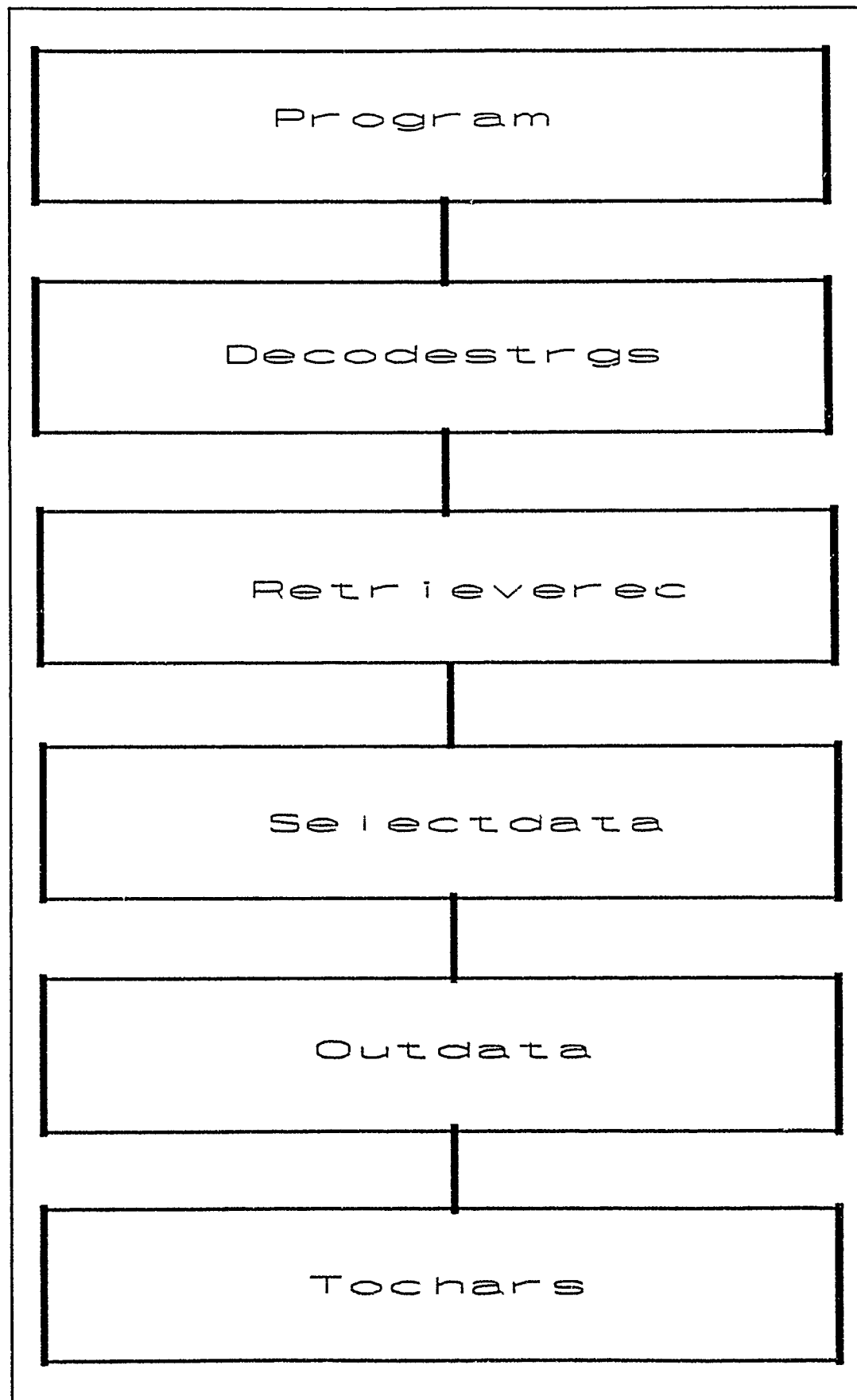
BEGIN
    CHOICE:=0;
    WHILE CHOICE<>3 DO
        BEGIN
            DISPMENU;
            READLN(CHOICE);
            CASE CHOICE OF
                1: DISPINDEX;
                2: DELENTY;
            END;
        END;
    END;
END.

```

A2 Program Structure Charts

DRTSTAT.PAS





A3 Sample Program Outputs

Output from DRT1.COM

DRT STATISTICS PROGRAM

Enter filename: Enter filename: CND3139

Control File: CTL3139

Source: Multi-Listener Master V4
Date: 9-NOV-89

Input Device: B&K WB TAPE
Input Noise: NONE
Link Device: SPECIAL CODER
Output Device: BT HANDSET
Output Noise: 65DB(A)USASI

! Optional Header Text
HANDSET TEST
TAPE 8
Sequence Type: DYNA
Filler Display: Enabled

Words File: DYNABRD
Sequences File: DYNASEQ

DRTs: 10

! Talker	List
RLP	306A.1
RLP	306A.2
RB	308A.1
RB	308A.2
MCL	315A.1
MCL	315A.2
PRW	312A.1
PRW	312A.2
GAP	314A.1
GAP	314A.2

Listeners: 10

! Listener	Station
ABBOTT	1
BERRY	2
BLACK	3
MCINTOSH	4
RUSBY	5
HOOK	6
JON1	7
TAYLOR	8
JON2	9
SLATER	10

TUKEY TEST FOR NONADDATIVITY

Source	Sum of Squares	Degrees of Freedom	Mean Square
LISNERS	477.09	9	53.01
TALKERS	624.88	4	156.22
RESIDUAL	460.19	36	
NONADD	3.38	1	3.38
BALANCE	456.81	35	13.05

ANALYSIS OF VARIANCE

Source	Sum of Squares	Degrees of Freedom	Mean Square	F-Ratio
LISNERS	954.25	9	106.03	1.99
TALKERS	1249.75	4	312.44	5.87
LN * TK	920.38	36	25.57	0.48
ERROR	2660.56	50	53.21	
TOTAL	5784.94	99		

NEWMAN - KEULS TEST

LISTENERS :

2 3 8 4 6 1 10 9 7 5

TALKERS :

4 3 5 1 2

ATTRIBUTE SCORES -----

	Present		Absent		Mean	
	mean	SE	mean	SE	mean	SE
Voicing	87.5	2.5	70.5	4.7	79.0	3.2
Nasality	96.5	0.6	84.3	3.5	90.4	2.2
Sustention	72.8	2.8	69.3	2.8	71.0	2.0
Sibilation	82.5	1.9	81.5	2.2	82.0	1.4
Graveness	61.0	2.4	62.3	2.8	61.6	1.8
Compactness	90.0	2.6	80.5	3.9	85.3	2.5

INDIVIDUAL SCORES -----

LNR	T1	T2	T3	T4	T5	SD
1	79.2	82.3	72.9	76.0	86.5	5.3
2	76.0	80.2	67.7	61.5	71.9	7.3
3	82.3	78.1	72.9	70.8	71.9	4.9
4	77.1	81.3	76.0	80.2	76.0	2.4
5	84.4	86.5	76.0	78.1	86.5	4.9
6	83.3	82.3	76.0	77.1	78.1	3.2
7	84.4	85.4	77.1	68.8	88.5	8.0
8	75.0	80.2	79.2	65.6	77.1	5.8
9	83.3	81.3	76.0	76.0	85.4	4.3
10	81.3	83.3	76.0	76.0	81.3	3.3

LNR	MEAN OVER TKR
1	79.4
2	71.5
3	75.2
4	78.1
5	82.3
6	79.4
7	80.8
8	75.4
9	80.4
10	79.6

STD. DEVN. = 3.3
STD. ERROR = 1.0

TKR	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	SD
1	79.2	76.0	82.3	77.1	84.4	83.3	84.4	75.0	83.3	81.3	3.5
2	82.3	80.2	78.1	81.3	86.5	82.3	85.4	80.2	81.3	83.3	2.5
3	72.9	67.7	72.9	76.0	76.0	76.0	77.1	79.2	76.0	76.0	3.1
4	76.0	61.5	70.8	80.2	78.1	77.1	68.8	65.6	76.0	76.0	6.1
5	86.5	71.9	71.9	76.0	86.5	78.1	88.5	77.1	85.4	81.3	6.2

TKR	MEAN OVER LNR
1	80.6
2	82.1
3	75.0
4	73.0
5	80.3

STD. DEVN. = 4.0

STD. ERROR = 1.8

DRT SCORE = 78.2
STANDARD ERROR = 1.0

Output from DRT2.COM (written to DRTTMPFL2.DAT)

5

: "DRT " "INDIV " "POSIT " "ATTR" "COND"

TITLE

62.5	14	1	1	1
75.0	14	1	2	1
75.0	14	1	3	1

etc...

75.0	15	2	4	4
46.9	15	2	5	4
75.0	15	2	6	4
-1	-1	-1	-1	-1
-1				

Output from DRT3.COM

CCF> @drt3

DRTEDIT

1. Display Index
2. Delete an entry
3. End

Select.. 1

	1191	1192	1193	1194	1195	1196	1197	1199	1200	1201	1202
1203	1204	1205	1206	1207	1208	1219	1220	1221	1222	1223	1224
1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236
1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248
1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260
1261	1262	1263	1264	1265	1266						

DRTEDIT

1. Display Index
2. Delete an entry
3. End

Select.. 3

A4 Sample Control File for DRT2

Example Control File (CF....)

```
! This file is for EXAMPLE purposes only, and will not run.
! Lines starting with ! are used for comment here, but will cause fatal
errors at runtime.
!
! Make the control file name the same as the filename used to store
this file
!
! N.B.!!! The "star" symbols must separate the different entries.
!
Control File: CFwhatever
*
!
! These numbers after the colon must tally with the number of lines
below them.
!
Attributes:1
TKR
*
! List CND numbers here. These are in the order corresponding to the
! Design statement used in PANOVUN.
!
Conditions:24
1338
1349
1342
1353
1335
1348
1334
1345
1331
1344
1339
1352
1337
1350
1341
1354
1336
1347
1333
1346
1332
1343
1340
1351
*
!
! Select listeners whose results you wish to use.
! Some degree of " missing values" (less than 10%) is acceptable
```

! YOU MUST LEAVE TWO SPACES in front of the four letter name.

!

Listeners:12

CHAM

PRAT

WHEA

SLAT

PAWS

BERR

FERR

PERK

SIMO

KNUT

TAYL

PLOW

*

!

! First line is ALWAYS "Format:4"

! Second line is the number of variables.

! Third line is the nesting of the variables.

! Fourth line is the quantity of each of the variables.

! Fith line is a title that will appear on the printout.

!

Format:4

5

"TALKER","REPLIC","UTTER ","CODER ","SNR "

5 2 2 3 2

Title line

*

!

! These statements drive PANOVUN, an analysis of variance program

! Written by the Institute of Aviation medecine.

!

Anovar Design:-

Another title

Y

1

N

:E

1

E

D Put your design statement here

A(IY,V=-999)DRT

E

Q

*

!

! Listener ident. numbers are put here.

! Last listener is always "----"

! Further listeners may be added as required.

! BUT LEAVE TWO SPACES

!

!

!

Listener Idents:14

CHAM 1
PRAT 2
WHEA 3
SLAT 4
PAWS 5
BERR 6
FERR 7
PERK 8
SIMO 9
KNUT 10
TAYL 11
PLOW 12
ROBE 13
---- 14

*

REPORT DOCUMENTATION PAGE

DRIC Reference Number (if known)

Overall security classification of sheet UNCLASSIFIED.....
 (As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the field concerned must be marked to indicate the classification eg (R), (C) or (S).)

Originators Reference/Report No. REPORT 91026		Month AUGUST	Year 1987
Originators Name and Location RSRE, St Andrews Road Malvern, Worcs WR14 3PS			
Monitoring Agency Name and Location			
Title DIAGNOSTIC RHYME TEST STATISTICAL ANALYSIS PROGRAMS			
Report Security Classification UNCLASSIFIED		Title Classification (U, R, C or S) U	
Foreign Language Title (In the case of translations)			
Conference Details			
Agency Reference		Contract Number and Period	
Project Number		Other References	
Authors SIM, A; BAIN, R; BELYAVIN, A J; PRATT, R L.			Pagination and Ref 59
Abstract This report describes the statistical techniques and associated computer programs employed to analyse data obtained from Diagnostic Rhyme Tests (DRT), and was previously published as a Divisional Memorandum. The conduct of the DRT (used for quantifying speech intelligibility) was described in RSRE Report No 87003.			
			Abstract Classification (U,R,C or S) U
Descriptors			
Distribution Statement (Enter any limitations on the distribution of the document) UNLIMITED			
S8Q/48			